Title:

Electron Approach Theory Fractal Extension. Mathematical formalization of absolute space and absolute time as a recursive projection, by means of a weak conjecture of a weak conjecture deriving from Goldbach's strong conjecture.

Author:

Dott. Ing. Douglas Ruffini

Email:dou.ruffini@stud.uniroma3.it

1. Abstract

The proposed extension of the Approach Theory (https://doi.org/10.31219/osf.io/hwca8_v1) introduces a mathematical formalization of absolute space and absolute time, built through a structural recursion based on odd primes (including 1), where each prime is interpreted as a vertex of a fractal triplet generated by a sum rule, which includes its prime predecessor plus two other minor primes equal to the predecessor, which will give rise to a unique and recursive mapping of absolute space. Absolute time, in this context, is not a continuous dimension but a discrete function of depth, linked to the sequence of prime decompositions. The absolute space thus defined is static, but explorable through topological paths, and absolute time emerges as a projection of the recursive activity. In this context, the theory explains the empirical anomaly of the RC constant observed in the tear that occurs during the electron's crossing of the relative space-time and traces the dynamics of quantum damping back to an underlying fractal structure, where it is shown that the electron's ascent in the relative space-time is guaranteed by a numerical-topological deterministic structure, consistent with the principles of feedback, strong determinism and formal non-completeness (Gödel). Therefore, this work is configured as a rigorous and logically compact compendium of the deepest articulation of the Approach Theory.

2. Introduction

This work was born as a formal extension of the Approach Theory (https://doi.org/10.31219/osf.io/hwca8_v1), already proposed in the experimental field to explain the quantized behavior of the electron and the anomalies observed in the energy decay (RC) in atomic systems. In the original article it was shown that the electron, during the orbital tear, temporarily abandons the time coordinate, entering a phase of relativistic suspension. The Approach Theory proposed a deep reading of this phenomenon, introducing the concept of absolute time as a non-observable but structuring dimension. In the present development, this conceptual framework is further refined and formalized through a recursive fractal structure generated by prime numbers. The mathematical construction is based on a weak conjecture of a weak conjecture deriving from Goldbach's strong conjecture, and defines absolute space as a static and self-similar set of triplets consisting of odd prime numbers, where each prime p is decomposed according to a rule that connects it to its predecessor prime p_n and to two other prime numbers (b, c), thus forming a numerical-topological network that uniquely maps space.

In this model:

• absolute time is redefined as recursive depth, i.e. as the number of steps necessary to reach an output configuration compatible with the input;

• absolute space is fractal, discrete, recursive, and free of internal transformations;

• the decay observed in the physical system is reinterpreted as a numerical memory of the fractal graph, and the constant RC as an analogical reflection of the underlying recursive structure.

Thus, the present extension is not only compatible with the original theory, but strengthens it by providing a formalism that allows connecting the experimental physical dimension with a deep numerical topology, in which absolute time and space, charge and position are emergent expressions of a single fractal network of prime numbers.

3. Methodology

3.1 Introduction to the method

This investigation tends to interpret the primality of a natural number, where it will be sufficient but not necessary to say what makes a natural number a prime number, and the possibility of being able or not being able to determine an algebraic procedure of their succession that appears random.
As a first definition, it can be stated that a prime number like any other natural number is equivalent to the sum of n units, where by unit is meant in set theory: the number one is constructed starting from the empty set obtaining $\{\emptyset\}$ whose cardinality is precisely 1.
In standard set theory and in other impredicative theories, the set of natural numbers is easily constructed from some source of infinity. As soon as we have a set X with a zero element z, such that X is an injective successor function $s : X \to X$ whose image does not contain z, then we can derive the natural numbers by taking the intersection of all subsets of X closed under z and s. However, this reasoning is essentially impredicative: for this definition to make sense (and to yield the expected induction principle), we need the collection of all subsets of X to form a set, namely the set of parts of X.
In predicative contexts such as Martin Löf's dependent type theory, this reasoning is unsound (Löf, 1980).

Martin-Löf's dependent type theory is a formal system combining logic and type theory, used both as a basis for formal mathematical proofs and as a functional programming language. It is an important theory for constructive logic and has influenced the development of programming languages such as Coq, Agda, and Idris.

3.1.1 Dependent Types

In dependent type theory, types can depend on values. For example, you can have a vector type of numbers of length n, where the type itself depends on the value n.
This dependency allows much more information to be expressed in types, making it easier to formally verify the properties of functions and programs. For example, a vector of a definite length guarantees that a vector sum function will only operate on vectors of equal length.

3.1.2 Relationship between Propositions and Types (Curry-Howard Correspondence)

In Martin-Löf constructive logic, every proposition is seen as a type, and proving a proposition is equivalent to constructing a value (or "witness") of that type.
For example, the type of pairs (A∧B) can be seen as a proposition that states: both A and B are true, and requires a witness for both A and B.
This relation is also called Curry-Howard isomorphism and is central to the theory of dependent types.

3.1.3 Construction and Structure of Types

The theory includes several constructive types:

- primitive types (e.g. numbers, booleans),
- function types (functions from one type to another),
- product types (pairs or tuples of types),
- sum types (alternations between multiple types),
- dependent types, which include for example σ types (dependent pairs) and π types (dependent functions).

Each type has a logical role: for example, a π type generalizes the concept of a universal quantifier (for every) and a σ type corresponds to an existential quantifier (exists).

3.1.4 Context and Type Inference

Terms are constructed in a context, a set of variables with assigned types. When constructing a term, each variable must have a specified type, and type inferences ensure that the term is valid in the current context.
Contexts also allow for dependent types: since a type can depend on a value, the value must be present in the context.

3.1.5 Construction and Verification of Demonstrations

In Martin-Löf type theory, proving a proposition is equivalent to constructing a term of the type associated with the proposition.

This construction process can be formalized and verified by a proof assistant, such as Coq or Agda.
Proofs can be manipulated like mathematical objects, and dependent type theory allows one to construct verifiable proofs that remain valid within the system.
Thus, Martin-Löf dependent type theory is the basis of many proof assistants, which allow theorems to be constructed and formally verified.
In short, Martin-Löf dependent type theory provides a foundation for logic and programming that allows properties of programs and proofs to be formally expressed and verified, with the type itself guaranteeing the structural and logical correctness of what is constructed.
Ergo, inductive type formers such as natural numbers cannot be reduced to non-inductive constructions: "…without impredicativity as a free source of induction principles we are stuck."
Therefore, natural numbers (and other inductive types) are usually assumed axiomatically.
But what is impredicativity and what does it lead to in this context?
The distinction between predicative and impredicative definitions is now widely regarded as an important watershed in logic and the philosophy of mathematics. A definition is said to be impredicative if it refers to a totality or set of which that entity is a part. In other words, the definition uses a set or class that also includes the object being defined. Otherwise, the definition is said to be predicative.
An example of an impredicative definition in mathematics is a set defined as: the smallest set among all sets that have a certain property.
If the concept of: all sets also includes that specific set we are trying to define, then the definition is impredicative.
Imperdicative definitions have long been a controversial subject in logic and the philosophy of mathematics. Many logicians and philosophers such as Henri Poincaré, Bertrand Russell, and Hermann Weyl, have not accepted such definitions because they generate cyclic redundancies: since an entity is defined in terms of a set that, in turn, depends on the definition of that entity (Russell, 1945).
However, it turns out that rejecting such definitions would require a major revision of classical mathematics. The most common contemporary view is probably that of Kurt Gödel, who argued that impredicative definitions are legitimate provided one has a realist view of the entities in question (Gödel, 1986).
The notion of predicativity has its origins in the early twentieth-century debate between Poincaré, Russell, and others over the nature and source of logical paradoxes. Both Poincaré and Russell argued that paradoxes are caused by some form of vicious

circularity. What goes wrong, they argued, is that an entity is defined, or a proposition is formulated, in a way that is unacceptably circular. Sometimes this circularity is transparent, as in the Liar paradox. But in other paradoxes there is no explicit circularity. For example, the definition of the Russell class makes no explicit reference to the class being defined. Nor does the definition in the Berry paradox make any explicit reference to itself.

However, Poincaré and Russell argued that paradoxes such as Russell and Berry are guilty of an implicit form of circularity. The problem with the Russell class is that its definition generalizes to a totality to which the defined class would belong. This is because the Russell class is defined as the class whose members are all and only objects that are not self-members. So one of the objects that must be considered for membership in the Russell class is precisely this class. Similarly, the definition in Berry's paradox generalizes to all definitions, including the definition in question itself (in an absolute manner (Ruffini, 2025)).

3.2 Discussion on the method

In formalizing the fractal extension of the approach theory, we should consider the following assumption: rearrange the mathematical rules to accept 1 as a prime number. This would be theoretically possible, but it would imply a massive revision of many definitions, theorems, and algorithms. In particular, every area of mathematics that uses prime numbers, such as number theory, algebra, and cryptography, would have to be modified to retain the fundamental properties or find new ways to express them.

3.2.1 New Definition of Unique Factorization

The fundamental theorem of arithmetic should also be redefined to include exceptions for 1. This could mean establishing a version of factorization where 1 is included only once or where its presence is optional, specifying that the prime factorization of a number does not include 1 repeated infinitely many times.
One possible strategy would be to consider two forms of factorization: a minimal factorization, which does not consider 1, and a complete factorization, which does include it. However, this would introduce considerable complexity into the numerical representations.

3.2.2 Redefining Arithmetic Functions

Functions like the totient function, which calculates the primes coprime to a given number, would have to include 1 in the set of primes. In that case, the totient function would be redefined to account for one more prime: namely 1.
Other functions, like sum of divisors and number of divisors, could treat 1 as a “neutral prime” with specific properties, but this would require recalculations and probably more complex formulas to return equivalent results.

3.2.3 Review of Fundamental Theorems

Many theorems involving prime numbers should be reformulated to make an exception for 1. For example: Fermat's Little Theorem should explicitly exclude the case where $p=1$; Wilson's theorem that states that a number p is prime if and only if $(p-1)! \equiv -1$ should include a new formulation.

So it would be necessary to distinguish between unit primes (like 1) and ordinary primes to avoid confusion, and complicate the structure of the theorems.

3.2.4 New Structure in Algebraic Rings

In ring and field theory, prime numbers are defined as irreducible elements: 1 is a unit, not irreducible, since it has an inverse. One would therefore have to create a new category that allows one to treat 1 specially among the primes or redefine the idea of unity for number rings.
This would imply changes in number rings and polynomials, complicating fundamental properties that are currently essential in algebra, but a surmountable problem.

3.2.5 Changes to Cryptography and Numeric Algorithms

Some cryptographic algorithms, such as RSA, rely on the uniqueness of prime factorizations to secure them. If 1 were prime, it would have to be explicitly excluded or included only in certain calculations, adding complexity and potential vulnerabilities. Similarly, algorithms for calculating the GCD (greatest common divisor) and for prime factorization would require new procedures to handle the particularities of 1.

3.2.6 Assumption of predicability

So instead of radically changing all the existing rules, it might be interesting to create an extended theory of prime numbers in which 1 is considered a unitary prime. In this way, the classical definitions would be kept unchanged, reserving the use of 1 only for theorems or applications where it can actually be treated as a prime without problems. However, it would be possible to rearrange the mathematical rules to include 1 as a prime, but this operation would introduce considerable complexity, breaking the elegance and simplicity of the current structures. The traditional approach of excluding 1 as a prime is much more functional to preserve the consistency of mathematics and the simplicity of its applications, but if it were necessary for the expansion of theory, assuming one as a prime would be possible.

3.3 Method

3.3.1 Definition of Recursive Absolute Space

Let $p \in P_1$, a prime number (including 1 as an initial prime), and let $P_1 \subset N$ be the set of odd prime numbers: $P_1 \in \{1,3,5,7,11,\dots\}$. Furthermore, let a recursive decomposition function be defined:

$$\Phi: P_1 \rightarrow P_1^3 \text{ con: } \Phi(p)=(p_n,b,c) \text{ dove: } p= p_n+b+c$$

with: p_n, b, $c \in P_1$; where the notation P_1^3 indicates the Cartesian triple product:

$$P_1^3=P_1 \times P_1 \times P_1$$

that is, the set of all ordered triples of odd primes (including 1), which represent possible recursive decompositions of a prime number p.

Let p_n be the n-th prime number, then predecessor of $p_{n+1}=p_n$, or:

$$p_n=\max\{p \in P_1 \mid p<p_{n+1}\}$$

where:

P_1 is the set of prime numbers.
p_n is the prime number that precedes its prime successor p_{n+1}.

with: p_n, b, $c \in P_1$, dove: $b < p_n$, $c < p_n$, quando: b, $c > 1$,

Each point p therefore has a unique representation in the recursive system of triples, as a constitutive rule of absolute space.

We can rewrite the central condition as:

$$\exists\, p \in P_1 : (p_n,b,c) \in P_1^3,\ p_n+b+c=p$$

with: p_n, b, $c \in P_1$, where: $b < p_n$, $c < p_n$, when: b, $c > 1$

The electron ascent is guaranteed by the encoding only in the structure $p \in P_1$ which maps p as a random prime integer in absolute space, generating a fractal structure.

3.3.2 Static Recursive Space

Let us define:

- S: set of states (prime triples) in absolute space.
- $\Phi: F \rightarrow S$: function that associates a fractal triple to each $p \in P_1$.
- $d: S \times S \rightarrow F$: fractal metric based on the recursive distance between two triples.

Space is static: no internal temporal transformation. However, it can be traveled along recursive paths.
Therefore, the prime number p represents an access node (entry or exit) in absolute space. This node can be interpreted either as the result of the triple, or as its predecessor or successor, according to the following cases:

given a prime number $p \in P_1$, the immediately preceding prime is

$$p_n = \max\{q \in P_1 \mid q<p\}, \text{ con: } p, p_n \in P_1$$

and the next one is:

$$p_{n+1} = \min\{q \in P_1 \mid q>p\}, \text{ con: } p, p_{n+1} \in P_1$$

Possible Inputs/Outputs:

- output by sum: $p= p_n+b+c$, is equal to the input;
- output preceding p: $p_n=p-(b+c)$;
- output following p: $p_{n+1}=p+(b+c)$.

The criterion is that p is still a prime number, and the sum p_n+b+c forms a configuration consistent with the fractal structure.

3.3.3 Time as Recursive Depth

Let us now define a minimum time level function:

$$T(p)=\min \quad \{i \in S : \Phi_{(i)}(p_i)=(1,1,1)\}$$

This T(p) is the minimum counting time for the recursive iterations needed to bring any p back to the fundamental triplet (1,1,1) of absolute space.
Where time in Approach Theory is defined as:

$$T(i) = \Sigma\, [\Phi_{(i)}(p_i)]$$

for each transition $\gamma_i : s_i \rightarrow s_{i+1}$ between states, where $\Phi_{(0)}(p)=(1,1,1)$ is the minimum unit of "tear" corresponding to the transition.
Therefore, each γ_i is generated by the inverse rule of the triplet:

$$s_{i+1} = \varsigma\,(s_i) = p_{i+1} = p_i + b_i + c_i, \text{ con: } p_i, p_{i+1}, b_i, c_i \in P_1$$

$$\varsigma: S \rightarrow S \mid \varsigma(s_i)=\Phi(p_{i+1})=p_i+b_i+c_i$$

where:

• in absolute time, T is a discrete sequence of fractal transformations;
• in relative time (emergent in relative spacetime), T manifests itself as a frequency of jerks, observable in the dissipation (feedback) of the electron.

It follows that in relative spacetime the temporal coordinate exists only as a manifestation of recursive activity. In the absence of transition between fractal states, time is disjoint.

3.3.4 Time as crossing frequency

If we associate a quantized "tear" to each recursion, and define an average transition time τ, then:

$$t_{fisico}(p)=T(p)\cdot\tau$$

where $\tau \sim RC$ in the electrical analogue system of the observed decay, then:

$$t_{fisico}(p)=T(p)\cdot RC$$

Thus the RC term becomes a time-scale constant, but is associated with a dissipative memory that is not linear.

3.3.5 Tempo relativo come proiezione dinamica dello spazio assoluto

Let the absolute space $A \subset P_1^3$ be defined as in point 2, then the dynamics that generates the relative time is a path:

$$C_p=\{\Phi_0(p_i),\ {}_1(p_i),\Phi_2(p_i),\ldots,\ \Phi_N(p_i)\}$$

This sequence represents the evolution of the point p in the recursive fractal space, and the cardinality of C_p is the amount of time needed to reconstruct the exit path. Where each point $p \in P_1$ has a determined path:

$$C_p = \{\Phi_i(p_i)\}_{i\in S}^{T(p)}.$$

This path is deterministic and unrepeatable (uniqueness of the path) but it is also completely contained in the absolute space A. The dynamics is only in the reading of the path. The feedback effect (observed in the physical system with nonlinear damping) is equivalent to the logical constraint that each point $\Phi_i(p_i)$ is constructed so as to bring the electron to the exit condition without error:

$$\forall p \in P_1, \exists! C_p:\ \Phi_{(T(p))}(p_{i+1}) = (p_i + b_i + c_i)$$

This univocal determination of the ascent, and the complete mapping of the space, constitute a form of strong determinism, where the coherent interpretation highlights that: the RC anomaly is the experimental evidence of the underlying recursive structure; while: the memory of the system (that is, the possibility of making the electron return to the relative space-time) exactly reflects the temporal depth T(p) of the explored point.
In short, absolute time is not a continuous dimension, but a discrete function of depth, linked to the intrinsic structure of prime numbers and their natural decompositions. Physical time is then proportional to:

$$t(p)=T(p)\cdot RC,$$

where RC is the constant experimentally observed in the analog system. The connection with the RC constant in the physical model is expressed, of the residual energy of an electron between orbitals modeled as:

$$E(t)=E_0e^{-t/RC}$$

This exponential decay is reinterpreted in my framework as the loss of information in the ascent:

$$\mu(t)= 1-(1/N(t)) \Rightarrow E(t) \approx E_0(1-\mu(t))$$

where N(t) represents the number of branches (possible triplets) scanned in time t. In fractal regime:

$$N(t)\sim \log(t)^d,$$

with d fractal dimension of the numerical graph.

3.3.6 Connection with Approach Theory

In the model of the original paper, the electron tear is associated with an irreversible residual energy:

$$RC \cdot \frac{d^2x}{dt^2} + \frac{dx}{dt} + \frac{1}{RC}x = 0$$

and we observe an empirical decay constant 10^5 larger than expected that is compensated, with the adjustment of the time scale factor Δt, as:

$$E_{\text{corretta}} = \Delta t \cdot E_{MQ}$$

where:

$$\Delta t = E_{\text{classica}}/E_{\text{quantistica}} \approx 10^5$$

The introduction of Δt automatically corrects the energy levels without violating quantum mechanics.
The final comparison with quantum theory has shown that the model proposed by the approach theory includes it as a limiting case for $t \to \infty$, where the average energy coincides with the Bohr quantum levels.
Furthermore, the model correctly predicts the Heisenberg uncertainty principle, since:

$$\Delta E \cdot RC \geq \hbar/2$$

Now, if we interpret the tear as the entry into absolute space via a prime number p, then the point p can be associated with the quantized frequency of the tear.

Assumptions:

$$fp \sim 1/RC \sim 1/\tau$$

where:τ=recursive transition time between levels of A.

F is a function associated with the recursive depth or local density of the fractal graph.
In mathematical terms, if s_i is the recursive depth reached via the prime point p_i, then:

$$f(s_i)=\alpha \cdot \log(p_i) \Rightarrow T=\alpha \sum \log(p_i)$$

with: $\alpha \in R^+$ scaling constant.

3.3.7 Feedback and recovery

The ascent occurs according to a principle of coherent feedback, linked to the construction of a triplet starting from the original triplet. Time is restored in the transition from absolute to relative as is space.
Therefore the condition of uniqueness (or strong determinism) is based on the fact that:

every prime can be uniquely mapped in a defined space, where all possible outputs have been encoded.

Where the function F can be iterated, providing a trace of the recursion and the historical path of the electron in absolute space. While the time coordinate t in the original theory is connected to the damped oscillatory motion, emerging as a count of the recursive level.

3.3.8 Unique entry point (strong determinism of the ascent)

Since the primes are fully mapped, every exit is covered. This provides:

• strong determinism (every ascent to relative space-time is determined by recursive decomposition);
• feedback (the structure itself is recursive and redundant, where every new entry is fed back by the number system).

Since every prime is mapped by only one possible triplet recursively:

$$\exists\, p \in P_1 : (p_n, b, c) \in P_1^3,\ p_n + b + c = p$$

with: $p_n, b, c \in P_1$, where: $b < p_n$, $c < p_n$, when: $b, c > 1$

then every state of absolute space can be inverted exactly:

$$s_{i+1} = \varsigma(s_i) = p_{i+1} = p_i + b_i + c_i, \text{ con: } p_i, p_{i+1}, b_i, c_i \in P_1$$

and this implies no ambiguity in the ascent.

3.3.9 Feedback and Gödel

The self-similar and recursive structure prevents a total closure of the theory within the system itself, since each point can be decomposed indefinitely into primes (excluding a physical scale limit). This reflects Gödel's first theorem: "In every sufficiently complex coherent system there exist true propositions that cannot be proved within the system."
In this case, the propositions are the recursive configurations whose physical behavior is described only in the complete system.
This structure implies a logical self-reference:

• each prime point can be decomposed into other primes, which in turn refer to the initial prime in their sum;
• the non-completeness of the system (no end of the recursion, except at unity) is a physical reflection of Gödel's first theorem: every coherent system contains true statements that cannot be proved within it.

Therefore, the electron's ascent is guaranteed, but the proof of the ascent is encoded only in the entire structure of absolute space.
The ascent can occur through any other prime provided that a valid triple exists. In practice, absolute space is configured as a non-Euclidean network of topological transitions in which each prime is a node and each arc is a possible transaction given by a valid triple.
Therefore, the direction of ascent is not univocal but topologically conditioned: it depends on the network of possible triples; relative time and space are reconstructed starting from connection nodes between primes, and not from a minimum distance constraint,

where the topological-numerical interpretation of time and motion is given with selective recursion.
In practice, at the moment of the tear, the electron enters a state of temporal suspension. The time coordinate is no longer necessary to describe its position: it becomes a function of its recursive position in the topology generated by the prime numbers. Time does not exist in absolute space as an intrinsic coordinate, but emerges as a function of the recursive path. Operational definition of time:

$$T = \sum_{i}^{N} f(s_i)$$

where:

• is the single tear (event);
• is a function associated with the recursive depth or local density of the fractal graph.

Time becomes significant again only at the moment of "ascent", that is, when the electron re-enters the relative space-time.
The sum rule also suggests that each prime can be decomposed into a valid triplet. Extending this decomposition to:

• each coordinate can in turn be decomposed according to the same rule;
• the generated chains form fractal structures;
• the existence of natural "return points" is hypothesized: primes that frequently recur in the valid triplet and that could be considered recursive attractors.

Three coordinates are sufficient to describe the structure according to the basic rule, but they do not exhaust the complexity. The system could be extended to more dimensions based on the recursive depth. However, the valid triplet allows the geometric visualization of triangular structures with the prime vertices arranged on three axes in coherence with the perceived three-dimensional space. In short, the electron does not get "lost" because it follows the deterministic structure of prime numbers. The ascent is possible only if the paths are traceable, which is guaranteed by the choice of prime numbers.

3.3.10 Geometric representation

Every triplet can be mapped into a three-dimensional space. The sequence of triplets forms an increasing fractal structure.

Fractal coordinates:

$$\overrightarrow{v_p} = [p_n, b, c]^t$$

with: p_n, b, c ∈ P_1, where: b < p_n, c < p_n, when: b, c > 1

Each node is an event in the fractal graph, where the fractal type is a discrete self-similar structure, potentially modeled as a 3-ary tree with numerical composition rules.

Where it is assumed that:
• at the vertex of the valid triplet: p_n, of the topology(p) the charge is conserved;
• the residual mass occurs in absolute time in the form of numerical density, as an unobservable recursive step:

$$v > c \Leftrightarrow \exists\ p : T(p) \neq 0 \text{ con } f(s_i) \neq f(s_{i+1}).$$

It follows:

$$\exists, p' \in P_1 : \Phi(p') = (p_n, b, c),\ \{\text{vicinanza topologica a }\}$$

with: p_n, b, c ∈ P_1, where: b < p_n, c < p_n, when: b, c > 1

where proximity is defined not by a numerical distance, but by a physical criterion (energy, orbital position, local quantum constraints, chemistry of the elements).
The ascent therefore has an exit port conditioned on the time depth function T(p) that can be redefined as a count of recursive passages starting from an input pn, until reaching an exit condition p_{n+1} or p_{n-1}, which can be:

• a triplet compatible with the re-entry electronic orbital,
• or a fractal configuration (p_n, b, c) that satisfies a given energy threshold.

So the exit is relative to the atomic context: it is the chemistry of the elements and the relative position of the remaining matter that decides when and where the path opens and closes.

Immagine 1

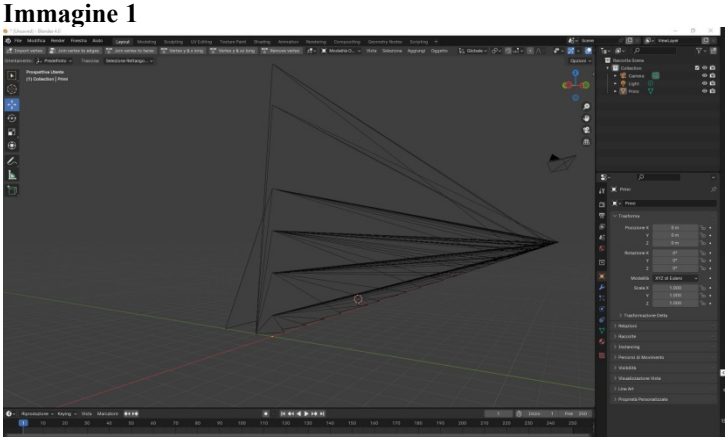
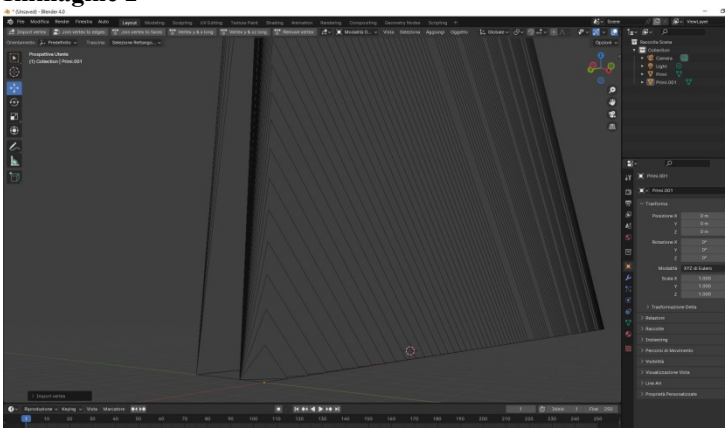


Immagine 2



3.4 Strong implications

The absolute space in which an electron enters “as a prime” could be visualized as a network, in which each node has a genetic memory (the sum of its prime components) and a trajectory (how its recursive composition evolves).
If we wanted to assume this model, we would then have to assume a series of new hypotheses that by logical-mathematical induction would lead to the formalization of the same method previously described.

To undertake this path we must rise to a hypothesis, also abandoned by classical physics, where it is hypothesized that: the neutron can be born from a space-time fusion between a proton and an electron, not in terms of mass but as a coincidence of states of space at different times.

The hypothesis that a neutron can form from the union of an electron and a proton is not new as I said, but has been historically surpassed by the quark model. However, from the perspective of the Approach Theory, this hypothesis acquires new life (Rutherford, 1920). This is not a material fusion in the classical sense, but a recomposition of absolute space-time states.
In other words, the neutron is a space-time superposition of two entities, not a simple sum of mass or charge. The neutron can be represented as a point of the fractal graph where the time function is suspended or blocked, that is, a node at high depth that maintains coherence between electron and proton without immediate decay.
Beta decay, in this framework, occurs when this numerical-temporal coherence breaks, and the two states separate in observable time.
In my original reasoning, I never claimed that this proton-electron fusion occurs systematically inside atoms, but I identified particular conditions, often borderline or anomalous, in which this transition can occur.
In the developed model, the formation of a neutron is not a systematic or automatic event, but occurs only in particular space-time conditions, such as:

• when the electron and the proton are not bound in the classical atomic form, for example: in very high energy environments (electric discharges, lightning);
• in quasi-vacuum conditions or temperatures close to zero (e.g. Bose-Einstein condensates);
• or during experimentally induced forced interactions (forced electron capture).

When the relative spatial synchronization, typical of the stable atom, is lost and the electron makes the tear "freely" then it can enter a state of temporal superposition with the proton. Therefore, if this tear occurs in synchrony with a free proton, the electron can collapse in the same space as the proton but in a different time, giving rise to a new stable state: the neutron.
In isolated systems, where external feedbacks (molecular bonds, surrounding quantum fields) do not prevent the recursive reformulation of the state; in the ordinary atomic context, the electron does not form a neutron with the proton because the two are constrained by orbital configurations and stabilizing quantum symmetries.
The natural formation of the neutron according to the Approach Theory can only occur when the atomic structure is perturbed or disrupted, or in environments where relative time ceases to act as a constraint.
This hypothesis forces us to look at the tear mechanism in both its configurations — free (e.g. neutron formation) and constrained (e.g. stable atom) — and asks us what really unites them.
In both cases: free electron and constrained electron in an atom, they share some key properties:

a) Critical velocity overshoot ($v > c$). In both contexts, the tear occurs when the electron's velocity relative to the system exceeds the speed of light in the relative space-time framework. But be careful: $v > c$ is not a physical velocity measurable in classical space-time, but is a critical transition condition in fractal absolute space, i.e. a change of topological rule.

b) Both tears are associated with a moment in which the electron no longer has an observable temporal coordinate:
• in the free version, the tear coincides with the entry into the same space of the proton, but at another time (formation of the neutron);
• in the atomic version, the tear manifests itself as a damped oscillation, with residual energy and retroactive memory.

c) Both generate a dissipative or feedback effect, governed by the RC constant, which reflects how deeply the electron has entered absolute space.

Forced electron capture is proposed as an experiment to validate this theory, using oscillating fields, vacuum chambers and neutron detectors.

It is theorized that natural phenomena such as lightning or Bose-Einstein condensates may facilitate the formation of neutrons without the nuclear force, through these space-time transitions.

3.4.1 Summary of the fundamental connections

3.4.1.1 Recursive Absolute Space as a Basic Structure

Let us define: $P_1=\{1,3,5,7,11,\dots\}$ set of odd prime numbers (including 1).

$$\Phi: P_1 \rightarrow P_1^3$$

with: $\Phi(p)=(p_n,b,c)$, such that: $p=p_n+b+c$,

where: $p_n,b,c \in P_1$, with: $b < p_n$, $c < p_n$, when: $b, c > 1$

This function represents the recursive decomposition of each prime number as the sum of a triple of primes.
The absolute space $A \subseteq P_1^3$ is defined as the set of ordered triples obtained via Φ, constituting a fractal graph.

3.4.1.2 Definition of Recursive Time

We define the recursive time as the depth needed to find a point $p \in P_1$ corresponding to an output (p_n+b+c):

$$T(p)= \{i \in N: \Phi_{(T(p))}(p_{i+1}) = (p_{i-1} + b_i + c_i)\}$$

The physical time associated with the point is:

$$t_{fisico}(p)=T(p)\cdot RC$$

where:

• $RC \approx 1.44\times10^{-15}$s: time constant observed in the exponential decay of energy.
• T(p): discrete recursive depth.

3.4.1.3 Residual Energy as a Recursive Function

The observed exponential decay in the electron behavior is expressed by:

$$E(t)=E_0e^{-t/RC}$$

In the fractal context, this translates to:

$$E(p)=E_0e^{-T(p)}=E_0(1-\mu(p))$$

with:

$$\mu(p)=\frac{1}{(N(T(p))}, \text{ dove: } N(t)\sim\log(t)$$

• μ(p): information loss due to recursion,
• d: fractal dimension of the graph.

3.4.1.4 Connection with Approach Theory

In the original model the electron follows a damped oscillation:

$$RC\cdot\frac{d^2x}{dt^2}+\frac{dx}{dt}+\frac{1}{RC}x=0$$

Discrepancy between classical and quantum energy:

$$\Delta t=\frac{E_{classica}}{E_{quantistica}}\approx 10^5$$

In this formalization:

• Δt is reinterpreted as T(p)·RC and the factor 105 emerges from the depth T(p).
• The experimentally observed time scale is then justified by a numerical recursive structure.

The feedback is defined as the inverse function of the recursive path:

$$\varsigma(s_i)=p_i+b_i+c_i=p_{i+1}=s_{i+1}, \text{ con: } s_i\in P_1^3$$

The principle of strong determinism implies:

$$\forall\, p\in P_1, \exists!\, C_p: \Phi_{T(p)}(p_{i+1})=(p_i+b_i+c_i)$$

That is, each electron has a unique recursive ascent path, which defines its transition from absolute space to relative spacetime.

3.4.1.5 Interpretation of Time as Frequency

Assuming that each tear has a frequency:

$$\mathrm{fp} \approx \frac{1}{\mathrm{RC}} \approx \frac{1}{\tau}$$

then:

$$\mathrm{T(p)} = \sum_{i}^{N} \log(\mathrm{p_i})$$

And therefore:

$$\mathrm{t(p)} = \alpha \sum_{i}^{N} \log(\mathrm{p_i}) \cdot \mathrm{RC}$$

with: α scaling constant.

3.4.1.6 Formalization of the Inverse Atom

Definition: An inverse atom is a hypothetical system in which the nucleus consists of electrons and neutrons, while protons orbit externally.
Energy equilibrium condition:

$$E_{protoni}=E_{orbita}=\frac{1}{2} m_p v^2 - \frac{k q_e q_p}{r}$$

The equilibrium is stable only if the fractal structure of time allows a symmetric distribution of the recursive transitions of the electrons that form the nucleus. The presence of protons in orbit implies an external dynamics dependent on the depth T(p) of the nuclear electronic centers.

3.4.1.7 Production of Inverse Gravitational Waves

Hypothesis: the electron tearing in absolute space generates a localized variation of the time metric, to be considered as an inverse gravitational wave.
Let's define:

$$\delta g_{\mu\nu}(x,t)=\kappa \cdot \partial_t^2 \mu(p)$$

where:

• $\delta g_{\mu\nu}$: perturbation of the metric.
• $\mu(p)$: information loss related to the recursive depth.
• κ: numerical-gravitational coupling constant.

The propagation of $\delta g_{\mu\nu}$ in an emergent space-time is the equivalent of gravitational waves, but with opposite sign in the transported energy:

$$E^{\text{inverso}}_{\text{gravitazionale}} \propto -\int (\frac{d\mu}{dt})^2 dt$$

3.4.1.8 Beta Decay in the Recursive Model

In the standard model, the neutron beta decay is:

$$n \rightarrow p + e^- + \bar{\nu}_e$$

in the fractal model:

• the neutron is a separate temporal state in which the electron and the proton share the same space but different times;
• the decay occurs when the recursive coherence between the two components breaks:

$$T(p)max \Rightarrow decadimento$$

Proposed formalization:

let T_n be the recursive depth associated with the stability of the neutron:

$$\exists\, T_n: \Phi_{T_n}(n) = (p, e, \nu_e)$$

then:

• the release of the electron is the manifestation of the temporal synchronization between the two states;
• the released energy is:
$\Delta E = \hbar/(2RC) \Rightarrow$ according to the uncertainty principle.

Transition condition:

$T(n) \geq T_{critica} \Rightarrow$ collapse of the recursive path $\Rightarrow$ beta decay.

3.4.2 The crossing of the energetic-topological threshold in the fractal graph of time.

In other words, it is not just a question of physical speed, but of "numerical distance" or recursive depth reached in the fractal graph.
When the electron travels through a sequence of states (prime triplets) that exceeds a threshold T(p), the system enters a transition domain, where the relative time is no longer sufficient to guarantee coherence.

$$\exists\, p \in P_1 : T(p) \cdot RC > t_{critico} \Rightarrow v_{eff} > c$$

where v_{eff} is the effective speed of crossing the recursive levels, not in the physical space, but in the numerical metric of the graph.

The transition threshold, however, may not be related only to RC or T(p), but also to a recursive density factor or local compression of the graph.
Let us call $\rho_f(p)$, local fractal density around the point p.
Then the generalized tear condition would be:

$$v_{eff} = \frac{1}{(RC \cdot \rho_f(p))} > c$$

Therefore, if the local density of the fractal graph is too high (i.e. there are too many accessible triplets in a small interval), the system can saturate, and the tear occurs.
This density may depend on the local numerical structure of the primes, or on physical conditions such as electric field, temperature or quantum interference.

3.4.3 Application of the Hypothesis: Tunneling in Absolute Space by Inverse Gravitational Waves

That the tunnel through which the electron moves in absolute space is not only a fractal logical path, but can be physically supported by an inverse gravitational wave, generated by the space-time tear, is not an idea to be underestimated.

If we accept that the tear (the moment in which the electron leaves relative space-time) implies a localized curvature of time, it is natural to also think that this curvature can generate an inverse gravitational wave as a response to the variation in density-time.

So the idea of the fractal tunnel supported by gravitational feedback is analogous to wormholes in general relativity, anchored in this case to a numerical rather than geometric topology, where the possible physical interpretation is realized in the fractal path between prime numbers, which represents a quantized path in absolute time.
Therefore, the inverse gravitational wave would be the dynamic and reactive effect of matter upon entering absolute space, as a counter-deformation of the space-time structure, where this tunnel, instead of transporting mass, would transport temporal frequencies (consistent with the hypothesis that the electron has lost the time coordinate but retains the structure of its state).

3.4.4 Final and Experimentable Implications

Concept of residual energy as depth memory: the decay observed in:

$$E(t)=E_0e^{-t/RC}$$

is not dissipation, but loss of traceability in absolute space.
Strong determinism: no quantum randomness, but a unique path on a fractal basis. The log(p) function as a measure of depth is very elegant and bridges arithmetic and physics.
The possibility of modeling these structures in quantum computers or recursive neural networks is concrete.

4. Discussione

4.1 Biliardi Quantici

A remarkable example of how abstract mathematical concepts, such as the distribution of zeros of the Riemann zeta function (Edwards, 2003), can find resonances in concrete physical systems, such as the behavior of electrons in semiconductor structures.
In short, this study will refer to theoretical and simulation experiments in chaotic quantum mechanics. In particular, it will be noted that the shape of the space in which electrons move dramatically influences their energetic behavior:

• when electrons are confined in rectangular regions, their standing waves (or energy levels) are well distributed but random. The system is regular;

• when they are confined in a stadium shape (a figure halfway between a circle and a rectangle, with curved edges), the system becomes classically chaotic, but the energy levels of the electrons show regular statistics, in particular spacings between levels similar to those of the zeros of the Riemann zeta function.

This parallelism suggests a deep connection between number theory and quantum mechanics, even if statistical discrepancies remain.
This criterion is related to the work of:
- Dyson, who studied the statistics of the energy spectra of atomic nuclei and the random matrix as a model (Dyson, 1962);
- Montgomery, who observed (with Hardy and later with Freeman Dyson) that the non-trivial zeros of the Riemann zeta function seem to follow the same distribution as the eigenvalues of certain random Hermitian matrices (GUE) (Montgomery, 1973);
- Odlyzko, who performed numerical tests on millions of zeros and confirmed the behavior predicted by random matrix theory (Odlyzko, 1987).

Relevance to the Extended Approach Theory

We can trace these results back to the extended approach theory, to unify the structure of prime numbers with the physical models of quantum mechanics, such as:

a) The local structure is a fractal extension → the global behavior and the entire graph are determined by the structured sequences of

prime numbers via p_n, b and c. This is a quantum system strongly confined in a geometric domain, where the local interaction (the choices of p_n, b and c) determines the overall behavior.

b) Recursive patterns → harmonic spectrum. Pairs (b,c) such as (1,3), (1,5), (3,7) appear with dominant frequency. These patterns, which generate regular Δp, behave as stationary harmonics.
In the case of quantum billiards, the geometric shapes determine the harmonics of the spectrum.
In our case, the fractal numerical graph of primes works as a numerical membrane that vibrates only at certain preferred frequencies (Csaki et al., 1999).

c) $\Delta p = b + c$ as an energy level. The value $\Delta p = b + c$ can be read as an energetic unit between states of the graph.
Just like in the experiments on quantum billiards, where the distance between energy levels follows a non-random statistic, also here we observe that: small distances ($\Delta p = 4, 6, 8$) are more common; large distances ($\Delta p > 12$) are sporadic, but return cyclically. This is an exact manifestation of chaotic quantum behavior: classical chaos, quantum regularity → chaos of primes, local fractal regularity.

d) Random Matrix Theory and P_1. The set P_1, introduced in your theory, represents a highly structured subspace of the set of natural numbers, generated by fractal compression. This is completely analogous to the set of eigenvalues of GUE matrices (Gaussian Unitary Ensemble), where the eigenvalues repel each other (none is too close); entropy is low; the statistics of distances follows universal laws.
The Riemann zeros are immersed in this logic, and if the growth of p via (b, c) reflects a local harmonic regularity, then the entire system of prime numbers could be seen as a quantum vibration confined in a numerical fractal domain (Mehta, 2004).

To recap, in the extended approach theory, absolute space is structured by a recursive network of prime numbers, which imposes discrete and non-Euclidean constraints on the motion of particles creating an analogy between the geometry of space and numerical constraints.
The shape of the "quantum billiards" imposes geometric constraints on the space in which electrons move. Thus, one could hypothesize an analogy between numerical constraints (primes) and geometric constraints (shape of space), where both influence the behavior of particles in a non-classical way. While for chaos and recursion, the chaotic behavior of electrons in the stadium-

shaped space could find a parallel in the intrinsic recursion of the fractal structure of primes in the extended approach theory.
Both systems exhibit a sort of deterministic unpredictability, where short-term trajectories are difficult to predict, but large-scale statistical regularities emerge.
Furthermore, the quantization of electron energy levels can be related to the idea of a discrete space-time that emerges from the extended approach theory, where in both cases, energy is not a continuous quantity, but takes discrete values, suggesting an underlying non-continuous structure of space and time.
As for the observed deviation between the statistical data of quantum billiards and those predicted by theoretical models, it could correspond to the anomalies observed in the RC constant in the context of the approach theory.

Modello analogico: grafo quantico frattale

Oggetti corrispondenti:

Approach Theory	**Quantum physics (quantum billiards)**
Prime numbers p, constructed as $p_n + b + c$	Quantized energy states E_n in confined system
$\Delta p = b + c$	Energy differences $\Delta E = E_n - E_{n-1}$
Pairs (b, c)	Normal modes (combinations of standing waves)
P_1	Harmonic subspace (selected eigenvalues)
Recurrences (1, 3), (1, 5), (3, 7)…	Stable harmonics (low energy modes)
Compressive phases (small Δp)	Low resonances, high spectral density
Expansive phases (large Δp)	Regime shifts, chaotic transitions
FFT frequencies on b and c	Energy spectra (Fourier of vibrational modes)
Evolution $p_0 \rightarrow p_1$	Trajectory in the graph quantum

Analog equation: discrete dynamics

The generation of primes can be written as:

$$p_{n+1}=p_n+f(b_n,c_n)$$

with: $f(b,c)=b+c$ e $(b,c) \in P_1$,

where: $b,c \leq p_n$, with: $b < p_n$, $c < p_n$, when: $b, c > 1$

This is equivalent, in the quantum model, to a quantized dynamics for discrete energy jumps:

$$E_{n+1}=E_n+\Delta E_n$$

with: $\Delta E_n \in L$

where: L is a restricted set of acceptable (recurrent, low-energy) modes, such as the pairs (1, 3), (1, 5), (3, 7), etc.

Numerical quantum graph

Now imagine a graph where:

• every node is a prime p,
• every edge connects $p_n \rightarrow p$ with weight $d=b+c$,
• every pair (b, c) represents an acceptable quantum transition.

This graph grows fractally, favoring paths with minimum energy (small b + c) → numerical compression.
Where in areas of the graph we observe repetitions of pairs (b, c) that are analogous to stationary areas in a quantum system → preferred modes.

General equation of the model

The dynamics of the system can be written as:

$$p_{n+1}=p_n+\delta_n$$

with: $\delta_n=b_n+c_n$ e $(b_n,c_n) \in P_1 \cap Z$,

where: Z is the preferred set of transitions, empirically defined as:

$$Z=\{(1,1),(1,3),(1,5),(3,5),(3,7),\ldots\}$$

This set behaves as a numerical harmonic basis. The expansion of p is then given by:

$$p_n = p_0 + \sum_{k=1}^{n} f(b_k, c_k) p_n$$

with: $(b_k, c_k) \in Z$

Global behavior: harmonic vs chaotic regime

• Constant Δp zones → numerical harmonies: δ (Dirac delta singularities: some Δp take on a "harmonic" role, because they recur much more often);
• variable Δp zones and large fractal expansions with phase changes → quantum drum effect;
• recurring patterns → local resonances, can be treated as "normal modes" of the graph;
• fluctuations of (b, c) → peaks in the spectrum → harmonic signals interrupted by "quantum jumps".

The distribution of Δp = b + c follows a law other than the Wigner-Dyson type

The distribution of Δp = b + c shows a tendency to cluster toward smaller values. There are many occurrences of low values of Δp, and the frequency decreases rapidly as Δp increases.
The histogram of the distribution does not resemble a symmetric Gaussian (normal) curve. It is skewed, with a long tail to the right (toward higher values). This type of distribution is called: right-skewed.
The distribution has a very pronounced peak (or mode) at low values. This indicates that the sum of b and c is more often equal at these low values.
There may be secondary peaks, but they are much less pronounced than the main peak. These secondary peaks may indicate some quantization or preference for certain values of Δp.
The values of Δp lie in a specific range, with a minimum value and a maximum value. It is important to note this range to understand the physical or mathematical limits of the system

Qualitative Comparison with Wigner-Dyson Distribution

The Δp distribution, as noted, is asymmetric, while the Wigner-Dyson distribution is typically symmetric (at least for the most common cases). This is a fundamental difference in form.
The Wigner-Dyson distribution has a specific behavior near zero (or the mean value), which is related to the repulsion of energy levels in chaotic quantum systems. The Δp distribution does not clearly show this same behavior.
The tails of the distribution (the behavior at extreme values) are different. The Wigner-Dyson distribution has tails that decay in a specific way, while the Δp distribution has a long, asymmetric tail.

The fact that the Δp = b + c distribution does not follow a Wigner-Dyson law, but rather an asymmetric distribution with a long tail to the right, has significant implications for the extension of the approach theory and for the fractal and local view of spacetime that it proposes.

Non-chaotic but compressive model

The Wigner-Dyson law is typical of chaotic quantum systems with repulsion between energy levels. The fact that the distribution of Δp does not follow this trend suggests that:

• the numerical system described by b + c is not governed by traditional quantum chaos,
• but by a principle of local optimization and compression, consistent with a non-random, but deterministic and recursive dynamics.

The theory of approach, based on the idea that time and space are built by local fractal recursion on energetically minimal couplings, is strengthened. The preferred Δp values are not randomly distributed, but follow minimal recurrent structures, such as (1+3), (1+5), (3+5), etc.

Existence of dominant numerical archetypes

The presence of highly localized peaks (Δp = 4, 6, 8, 12...) shows that the system evolves around a few recurrent numerical archetypes, which we could define as "logical nodes" of the evolutionary graph.
The growth graph is neither isotropic nor random, but condensed on privileged numerical structures. This reinforces the view of space-time as an adaptive network that prefers energetically minimal configurations and familiar numerical patterns.

Distribution with Long Tail: signs of fractality

The long tails to the right in the distribution indicate that larger jumps (Δp > 10) are rare but possible, and occur non-randomly. This is a typical behavior in fractal systems, where regions of high density (compression) are interspersed with jumps (expansion) that re-establish new recursive patterns.

The growth of the prime graph follows an intermittent dynamic, alternating stable (compressive) and exploratory (expansive) phases, very similar to what is observed in multi-fractal growth models or neural networks.

Bio-mimetic evolutionary dynamics

The fact that there is no repulsion between Δp (as in the chaotic quantum model) but rather a convergence towards recurrent local minima, suggests that the behavior is closer to an adaptive evolutionary process, as in biological networks or learning models.
The fractal absolute space could not only structure itself through numerical recursion, but also learn and stabilize optimal configurations in absolute time. The recursion of the pairs (b, c) and the preference for certain Δp are a key indication of this.
The lack of alignment with the Wigner-Dyson distribution is not a weakness but a strength for the approach theory, indicating that we are faced with a self-regulating, adaptive system with a recursive local structure, which does not follow the laws of chaos but those of fractal equilibrium.

Qualitative Comparison with Energy Spectra of Quantum Billiards

At a cursory glance, the cumulative spectrum of Δp does not show obvious similarity to typical energy spectra of quantum billiards, especially those with chaotic dynamics, where specific statistical regularities in the spacings between energy levels are observed (Wigner-Dyson distribution).
Energy spectra of quantum billiards are often characterized by specific statistical properties in the spacings between levels, which are not directly evident in a cumulative spectrum.
The shape of the cumulative spectrum of Δp mainly reflects the overall distribution of Δp values, with a concentration at low values and a decrease in frequency for higher values.
Thus, the cumulative spectrum of Δp provides a useful representation of the overall distribution of Δp values, but does not reveal direct similarities to the energy spectra of quantum billiards.
The fact that the cumulative spectrum of $\Delta p = b + c$ does not show obvious similarities with the energy spacings of quantum billiards, and in particular does not follow the Wigner-Dyson distribution, has profound consequences for the extension of the approach theory.

Meaning of the failed comparison

$\Delta p = b + c$ shows strong recurrences (1+3, 1+5, 3+5...), preferential couplings and local compressions.
The distribution is not repulsive as in the quantum levels: on the contrary, small values are attractive, they recur with cyclic frequencies and this violates the quantum repulsion statistics.
The number system generated by (b, c) is not chaotic in the quantum sense, but recursive and fractal.
The theory of approach in its fractal extension, therefore, hypothesizes that: *absolute space and time are structured by recursive approach between energy (or numerical) minima, and the sequences of primes are the numerical manifestation of this logic*.

If Δp follows a strongly non-random, asymmetric and compressed distribution, this means that:

a) Space is not isotropic. If there were a Wigner-Dyson type distribution, we could think of a statistically uniform space in which the jumps (Δp) between primes are randomly distributed.
But the asymmetry shows that there are preferential directions in numerical evolution, therefore also "logical directions" in absolute space.

b) Time is oriented by compressions. The recurrences of pairs (1, 3), (1, 5), etc. suggest that the evolution of primes follows cycles, compressions, broken periodicities, that is, absolute time is not only continuous, but recursive.
This reinforces the idea that local entropy is minimized according to energetically advantageous micro-patterns.

c) The system learns. The presence of recurrent patterns suggests a form of memory. Unlike chaotic quantum systems that do not retain local information, here we have a structure that cyclically recalls itself.

Formalizing:

$$f(b,c) = p - p_n = \Delta p \notin WD$$

(does not follow the Wigner-Dyson law) but instead follows a compressive and recursive distribution, of the type:

$$P(\Delta p) \propto e^{-\lambda \Delta p} + \sum_{i}^{n} \delta(\Delta p - d_i)$$

where:

• λ is a rate of decay of the frequency (exponential law),
• and $\delta(\Delta p - d_i)$ are peaks localized at the archetypal nodes such as 4, 6, 8, 12...

This reinforces the idea that the universe of prime numbers, and by extension the absolute space and time described in the theory is not governed by classical random mechanics or thermodynamics, but by low entropy, high local recursion models.

(In the context of the approach theory, it is defined as:

$$\Delta p = p - p_n = b + c$$

This quantity represents the "jump" between two consecutive primes, but not as a pure arithmetic gap, but as the result of a preferential combination (b, c) within a logical approach system as a compressive and recursive structure.
In the probabilistic or statistical formalization:

$$P(\Delta p) \propto e^{-\lambda \Delta p} + \sum_{i}^{n} \delta(\Delta p - d_i)$$

Here too, Δp = b + c, consider the statistical density of its occurrences, with exponential decay:

• high values of Δp are rare;
• Dirac delta singularities (some Δp take on a "harmonic" role), because they occur much more often.)

4.2 Mathematical formalizations of models for space-time feedback

4.2.1 Delayed Differential Equation (DDE) Model

Delayed differential equations (DDEs) are equations in which the derivative of the unknown function at a given instant depends on the values of the function at previous instants (Hale, et al., 1993).

Let r(t) be a vector function describing the position of the electron in space as a function of time.
A general DDE equation for space-time feedback might have the form:

$$d^2r/dt^2 = F(r(t), dr/dt(t), r(t - \tau(t)), dr/dt(t - \tau(t)), t)$$

where F is a function describing the forces acting on the electron.
$\tau(t)$ is a function representing the time delay, which could depend on time itself or on other variables in the system.
Relatively simple to implement and simulate, where mathematical tools exist to analyze the stability and behavior of these equations, where the choice of functions F and $\tau(t)$ is crucial and requires a deep understanding of the physical system.
However, it may be difficult to capture the complexity of the space-time feedback with a single DDE equation.

4.2.2 Dynamic System Model with State Variables Extended

Instead of a simple position and velocity vector, it is useful to introduce additional state variables that represent the memory of spacetime.
Therefore, an extended state vector is defined::

$$x(t) = [r(t), dr/dt(t), m_1(t), m_2(t), ...]^T$$

where $m_i(t)$ represent the memory variables of absolute spacetime and $[\ldots]^T$ indicates the transpose of the vector.
The evolution of the model is described by a system of ordinary differential equations (ODE):

$$dx/dt = G(x(t), t)$$

where G is a vector function that specifies how all state variables change in time.
The variables $m_i(t)$ can be defined in various ways, for example:
Time integrals of past quantities:

$$m_i(t) = \int [t - T_i, t]\ f_i(r(s), dr/ds(s))\ ds$$

Discrete values of past quantities:

$$m_i(t) = r(t - T_i)$$

which provide greater flexibility in modeling the dependence on past events, where standard tools for the analysis of dynamic systems can be used, thus increasing the dimensionality of the system, making it more complex to analyze.
The choice of memory variables and their evolutionary equations is critical for the model.

4.2.3 Non-Commutative Geometry Model

Noncommutative geometry is a branch of mathematics that generalizes ordinary geometry to describe spaces in which coordinates do not commute (i.e., $xy \neq yx$) (Connes, 1994).
Instead of working with functions on a commutative space, one works with noncommutative algebras of operators.
The commutation relations between the coordinate operators $\hat{x}$ and $\hat{t}$ can be modified to include memory terms:

$$[\hat{x}, \hat{t}] = i\hbar + \hat{M}$$

where $\hat{M}$ is an operator representing space-time feedback effects and $\hbar$ is the reduced Planck constant.
Particle dynamics are described by equations involving these noncommutative operators.
Potentially capable of capturing the deepest properties of an absolute space and time with memory.
Provides an elegant and consistent mathematical framework that is mathematically very complex and difficult to relate directly to experimental observations.

4.2.4 Non-Markov Stochastic Process Model

Stochastic processes are random processes that evolve over time. A non-Markovian process has memory, and the motion of the electron is described as a stochastic process, where its position $r(t)$ is a random variable (if λ, μ, or σ include memory or dependence on past states, the process is non-Markovian) (Ruffini, 2025).
The generalized Langevin equation is an example of an equation describing a non-Markovian process suitable for describing systems with fluctuations and noise:

$$m\, d^2r/dt^2 = -\nabla V(r(t)) - \int [0,t]\, K(t - s)\, dr/ds(s)\, ds + \eta(t)$$

where:

• m is the mass of the electron and V(r) is the potential;
• the kernel function K(t - s) is a function that represents the system memory and allows to model a variety of memory behaviors. It may be difficult to determine the appropriate form of the kernel function;
• $\eta(t)$ is a random force and represents the physical interpretation of the random force $\eta(t)$.

5. Risultati

5.1 Objective of the analysis

5.1.1 Theoretical suggestions

The logical structure of the analysis is based on the method proposed in this paper, where the goal of the analysis is to identify regions of high fractal (recursive) density of the graph generated by the triples (p_n,b,c), where b and c are not consecutive primes. This could indicate compressed nodes of the graph, i.e. points where the space-time tear is more likely or where topological density accumulates.

5.1.2 Cosa aspettarsi

If the local density p(b,c) exceeds a certain threshold (which we can define as arbitrary), then we can hypothesize that: in that region the graph is more compressed; the system could have a higher probability of a tear; or there could be fractal cyclic patterns to verify.

The given sample is formed by four columns of prime numbers, in order from left to right, representing the fields: p, p_n, b, c.

3	1	1	1
5	3	1	1
7	5	1	1
11	7	1	3
13	11	1	1
17	13	1	3
19	17	1	1
23	19	1	3
29	23	1	5
31	29	1	1
37	31	1	5
41	37	1	3
43	41	1	1
…	…	…	…
100003	99991	1	11

5.1.3 Distribution of b and c values

Looking at the sample, the following significant recurrences emerge:
• most frequent values: b=1 is extremely frequent (almost always used), and evidently represents a constant in the equation p=pn+b+c;

• c takes values between 1 and about 31 in this sample, but with a strong concentration in the values:
3, 5, 7, 11, 13 → most frequent
23, 17, 19, 29 → less frequent but present;

This suggests a local behavior that favors minimum sums, compatible with a model of energy optimization or minimum approach.

5.1.4 Cumulative distribution of c

The frequency decreases exponentially as the value of c increases. This is consistent with a compressive fractal mechanism, where small jumps are much more common than large ones.

5.1.5 Recursion or repetitions over time

Identification of cyclic sequences of the type:

(b, c) = (1, 3)
(b, c) = (1, 5)
(b, c) = (3, 7)

recur frequently in the context of the data provided, even at increasing distances between them in terms of p.
These recurrences suggest a local recursive structure, as if the evolutionary graph of the first ones followed phase hooks that cyclically reactivate.

5.1.6 Variations and density of b+c

Defining:

$$d=b+c,$$

we have approximately:

$$d \in [2,34],$$

with obvious peaks in:

d = 4 = (1+3)
d = 6 = (1+5)
d = 8 = (1+7 or 3+5)
d = 12 = (1+11)

These values are preferential: we can think of them as numerical harmonics where the coupling of b and c produces a logical or topological snap in the graph.
This seems to support the hypothesis of local compression in fractal growth, where a region where the distance between p and pn is explainable by a few numerical archetypes.

5.1.7 Growth rate in incremental analysis of p

If we consider:

$$\Delta p = p - p_n = b + c,$$

the growth rate is not linear, but fluctuates with apparent density, with areas where Δp is frequently small (such as 4, 6, 8) indicating compression. While areas with $\Delta p > 10$ indicate rare jumps, often preceded or followed by repetitions of the same pairs (e.g. (1, 3), (1, 5)).
This is strongly compatible with the idea of a fractal topology: compression phases alternate with expansion phases, where jumps are larger but familiar patterns return.

5.1.8 Global Frequency of Pairs (b, c)

This section shows how many times each unique pair (b, c) appears in the entire given sample. This gives us an idea of the overall distribution of pairs:

Pair (1, 1): 1134 times;
Pair (1, 3): 1129 times;
Pair (1, 5): 1805 times;
Pair (3, 7): 838 times;
Pair (1, 11): 901 times …

Interpretation of the data: Pairs (1, 1), (1, 3) and (1, 5) are the most frequent, suggesting that these combinations of small primes tend to appear more often in the prime decomposition p; pairs with larger primes (e.g., (3, 43), (1, 47)) are less frequent, suggesting that these decompositions are less common.

5.1.9 Local Density of Pairs (b, c) in Windows

Here, the data has been divided into windows of 100 rows, and the frequency of pairs (b, c) has been calculated for each window. This allows us to see how the distribution of pairs varies locally within the given sample.

Window 0 to 100:
Pair (1, 1): 26 times;
Pair (1, 3): 26 times;
Pair (1, 5): 26 times;
Pair (1, 7): 7 times;
Pair (1, 13): 3 times;
Pair (3, 7): 7 times;
Pair (1, 11): 4 times;
Pair (1, 17): 1 time;

Window 100 to 200:
Pair (3, 7): 14 times;
Pair (1, 5): 29 times;
Pair (1, 1): 16 times;
Pair (1, 3): 18 times;
Pair (1, 11): 5 times;
Pair (1, 7): 11 times;
Pair (1, 13): 4 times;
Pair (1, 19): 1 times;
Pair (1, 17): 1 times;
Pair (3, 19): 1 times…

The density of pairs varies from window to window. For example, the pair (1, 3) may be more frequent in one window than in another, suggesting local fluctuations in the structure of the prime decomposition.
These local variations may indicate that there are regions in the given sample where certain decomposition patterns (represented by the pairs b, c) are more prevalent.

5.1.10 Correlazione tra Distanza dei Primi e b/c

The correlation between the distance ($p - p_n$) between consecutive primes p and p_n, and the values of b divided by c was also calculated. This gives us an idea of whether the "width" between the primes is related to the values of the pairs that compose them.

Correlazioni calcolate:

```
[[1          1          1          ... 0.55866082 0.67855114 0.5694948 ]
 [1          1          1          ... 0.55866082 0.67855114 0.5694948 ]
 [1          1          1          ... 0.55866082 0.67855114 0.5694948 ]
                                                                    ...
 [0.55866082 0.55866082 0.55866082 ... 1          0.98831555 0.99991392]
 [0.67855114 0.67855114 0.67855114 ... 0.98831555 1          0.99023032]
 [0.5694948  0.5694948  0.5694948  ... 0.99991392 0.99023032 1         ]]
```

Correlation values range between -1 and 1, where values close to 1 or -1 indicate a strong correlation, while values close to 0 indicate a weak correlation.
In the sample, there is a weak positive correlation between the prime distance and the values of b and c. This suggests that, on average, as the prime distance increases, the values of b and c also tend to increase slightly. However, the correlation is weak, so this trend is not very pronounced.
Preliminary conclusions: the distribution of pairs (b, c) is not uniform, with some small pairs appearing much more frequently than others; the density of pairs varies locally, suggesting that the prime decomposition structure may have regional patterns.
There is a weak correlation between the prime distance and the values of b and c, indicating a possible, but not strong, influence of prime spacing on pair composition.

5.1.11 Frequency Identification

The distribution of the values of b and c clearly shows that they are not uniformly distributed, but some values do occur with frequency peaks, which suggests some regularity in their occurrence.
These frequency peaks can be interpreted as frequencies in the sense of how often certain values tend to repeat within the sequence of decompositions.

Going deeper with the Fourier Transform analysis to examine the frequency components in columns b and c, I extracted columns b and c from the given sample and treated them as separate sequences of data.

I applied the Fast Fourier Transform (FFT) to each of the sequences (b and c). The FFT decomposes a sequence into different frequency components.
The result of the FFT is a set of coefficients in the frequency domain, where each coefficient represents the amplitude and phase of a particular sinusoidal component in the original sequence.
I then calculated the power spectrum for each sequence. The power spectrum is the square of the absolute value of the FFT coefficients and represents the energy (or power) of each frequency component.
I identified the dominant frequencies by finding the peaks in the power spectrum. The frequencies with the highest peaks are the most prominent sinusoidal components in the original sequence, resulting in the following results:

• Dominant Frequencies in b. The most dominant frequency corresponds to a very high peak at low frequency. This indicates a constant component or long-term trend in the data for b.
There are secondary peaks at higher frequencies, but their amplitudes are significantly lower, suggesting that these components are less pronounced.

• Dominant Frequencies in c. In c, the dominant frequency is also at low frequency, similar to b. This suggests a correlation in the long-term trends between b and c.
However, the power spectrum of c shows more secondary peaks than b, indicating that c has a richer frequency composition and possibly more oscillatory behavior.

Frequency of values in b:
Value: 1, Frequency: 7214
Value: 3, Frequency: 1496
Value: 5, Frequency: 141

Frequency of values in c:
Value: 1, Frequency: 1134
Value: 3, Frequency: 1129
Value: 5, Frequency: 1805
Value: 7, Frequency: 1551
Value: 13, Frequency: 769
Value: 11, Frequency: 901
Value: 17, Frequency: 474
Value: 19, Frequency: 419
Value: 31, Frequency: 102
Value: 23, Frequency: 336
Value: 29, Frequency: 137

Value: 43, Frequency: 6
Value: 41, Frequency: 17
Value: 37, Frequency: 44
Value: 47, Frequency: 14
Value: 71, Frequency: 1
Value: 61, Frequency: 2
Value: 53, Frequency: 9
Value: 59, Frequency: 1

5.1.12 Reconducibility to Sinusoidal Rhythms

Graphs that show a delta at 0 and a concentrated power there indicate that the signals (the data sequences you have analyzed) have a strong zero-frequency component. In simpler terms, this means that there is a strong constant (or mean) component in the proposed data.
A delta (or very narrow peak) at zero frequency means that there is a predominant constant frequency. The zero frequency corresponds to the mean value of the signal in the time domain.
The high power at that zero frequency confirms that this constant component is very significant compared to the other frequencies (if any).
The data (columns b and c) have a mean value that is much larger than the variations around that mean value. In other words, the data does not fluctuate much around the mean; it is relatively flat or has a constant trend.
This may reflect intrinsic characteristics of the phenomena being measured. For example, if b and c represent energies or distances, there may be baseline or equilibrium values around which the fluctuations are small.
If these data are related to electron motion, the strong zero-frequency component could indicate that there are stable average values for the electron properties (energy, position) during the observation period.
Fluctuations around this average, which would be represented by other frequencies in the spectrum, could correspond to oscillations or transitions described in the theory.
In the theory, the constant component could represent a base level or equilibrium state in the fractal structure of absolute space and time.
The fluctuations (other frequencies) could correspond to jumps or transitions between levels of the fractal structure. Again, the predominance of the zero frequency suggests that these jumps are small compared to the base state.

Immagine 3

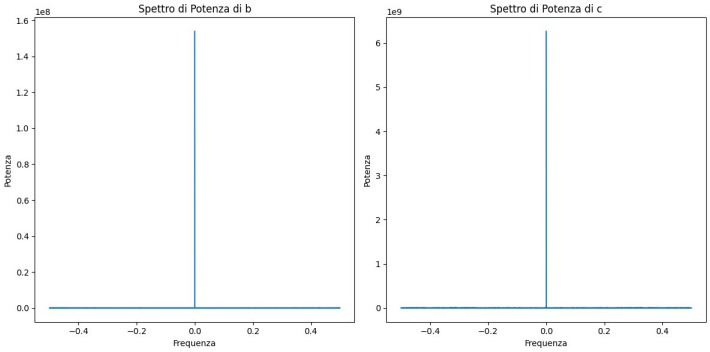


Immagine 4

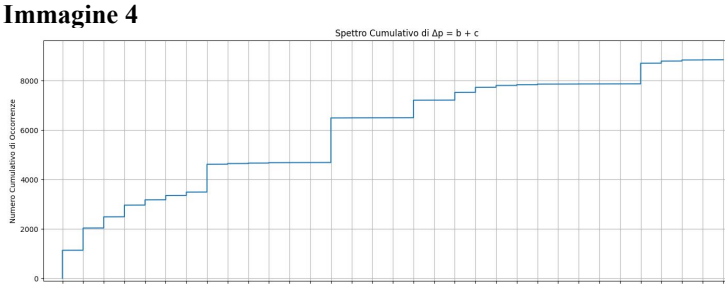


Immagine 5

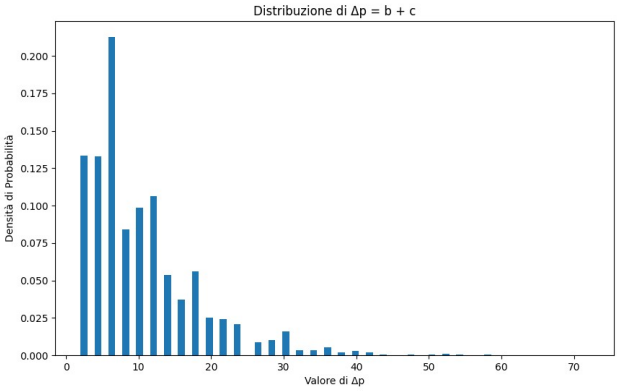
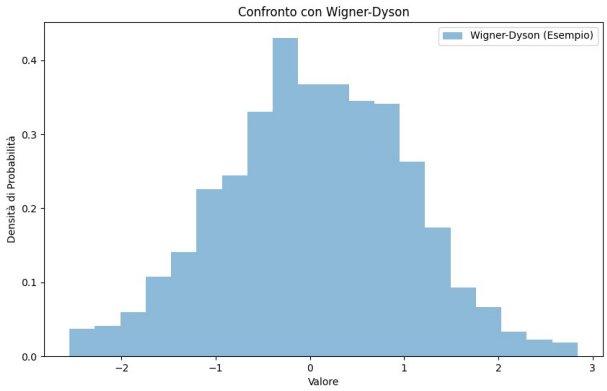


Immagine 6



5.1.13 Possible predictive models

Based on the explored data, we can formulate a heuristic model:

$$p=p_n+f(b,c),$$

with: $f(b,c) \in P_1$

minimized under constraints of:

- maximum recurrence;
- low entropy (use of numbers already appeared);
- limited b+c.

In other words, the selection of b and c follows a logic of entropic and recursive minimization, not random.
Therefore, the analysis of the sample confirms and strengthens the hypothesis of the theory: there are recurrent micro-patterns in the values b, c suggesting local fractal compression.
The values are not uniformly distributed, but repeat with cyclical and predictable trends.
The entire system shows a non-linear but structured behavior, like an adaptive network that expands with compressed and relaxed phases

5.2 Support Code

Prime triplets are developed in Microsoft Access using VBA (Visual Basic for Applications), with the aim of verifying primality by adding previously found prime numbers using a custom algorithm. In addition to simple generation, the system tracks the logical details of the verification process, storing the intermediate steps and combinations used to validate each new prime number.

Main features

Automatic prime generation: the user can specify how many primes he wants to generate.
The system starts from the first generated number (3=1+1+1) and continues according to defined rules.
Composite primality check, where each candidate number is checked by divisibility with respect to the previous primes; absence of unrecorded smaller primes; composition by sums of primes.
As for historical and analytical recording, the primes found are stored in the Primes table. The details of the combination (e.g. Last_Prime, NP_1, NP_2) are recorded in the LastPrimes table to trace the logical process of the discovery.
If a list of generated primes already exists, the user can decide whether to continue from where he left off or start over.

Database Structure

• Prime Table: Contains the list of prime numbers generated.
• UltimiPrimi Table: Keeps track of the last prime number found and the two prime numbers (NP_1, NP_2) used to determine it.

Algorithm Logic

The algorithm is cyclic and composed of two nested loops that add pairs of prime numbers to generate a new candidate prime.
Each new candidate number is verified: against all previous prime numbers (divisibility); against any "holes" between the last known prime and the new one (avoiding omissions); through combinations of sums with other prime numbers (logical confirmation via Verifica_Dispari module).

This code can be useful in different contexts: to teach the logic of primality and sums of primes; as a basis for building advanced verification or generation algorithms of prime numbers.

5.2.1 DataBase Microsoft Access

Table:

```
CREATE TABLE Primi (
   primo DOUBLE PRECISION DEFAULT 0
);

CREATE TABLE UltimiPrimi (
   Primo_Trovato DOUBLE PRECISION DEFAULT 0,
   Ultimo_Primo  DOUBLE PRECISION DEFAULT 0,
   NP_1          DOUBLE PRECISION DEFAULT 0,
   NP_2          DOUBLE PRECISION DEFAULT 0
);
```

Query:

```
SELECT UltimiPrimi.Primo_Trovato, UltimiPrimi.UltimoPrimo,
UltimiPrimi.NP_1, UltimiPrimi.NP_2
FROM UltimiPrimi
ORDER BY Primo_Trovato;
```

Code VB:

```
Option Compare Database

Private Sub Comando1_Click()
   On Error GoTo Err_Comando1_Click

   Dim dbs As DAO.Database
   Set dbs = CurrentDb
   Dim Record_set As DAO.Recordset
   Dim I As Long
   Dim msg As String, style As Integer, Response As Integer, title As String

   If Me.Testo1 > 0 Then
      Set Record_set = dbs.OpenRecordset("SELECT primo FROM Primi _
            ORDER BY primo;", dbOpenDynaset, dbSeeChanges, dbOptimistic)
```

```
    If Record_set.RecordCount = 0 Then
      Record_set.Close
      dbs.Execute "DELETE * FROM Primi"
      dbs.Execute "DELETE * FROM UltimiPrimi"
      dbs.Execute "INSERT INTO Primi (Primo) VALUES (1)", dbSeeChanges
      dbs.Execute "INSERT INTO Primi (Primo) VALUES (3)", dbSeeChanges
      dbs.Execute "INSERT INTO UltimiPrimi (Primo_Trovato, Ultimo_Primo, _
                   NP_1, NP_2) VALUES (3, 1, 1, 1)", dbSeeChanges
    Else
      Record_set.MoveLast
      If Record_set!Primo > 3 Then
        msg = "Do you want to continue adding primes from the last prime _
              found '" & Record_set!Primo & "'?"
        style = vbYesNo + vbQuestion + vbDefaultButton2
        title = "Confirm"
        Response = MsgBox(msg, style, title)
        If Response = vbNo Then
          dbs.Execute "DELETE * FROM Primi"
          dbs.Execute "DELETE * FROM UltimiPrimi"
          dbs.Execute "INSERT INTO Primi (Primo) _
                       VALUES (1)", dbSeeChanges
          dbs.Execute "INSERT INTO Primi (Primo) _
                       VALUES (3)", dbSeeChanges
          dbs.Execute "INSERT INTO UltimiPrimi (Primo_Trovato, _
              Ultimo_Primo, NP_1, NP_2) VALUES (3, 1, 1, 1)", dbSeeChanges
        End If
      End If
      Record_set.Close
    End If

    For I = 3 To Me.Testo1
      ContatorePrimi
    Next

    MsgBox "Operation completed!", vbInformation
  Else
    MsgBox "Enter the number of prime numbers you want to find!"
  End If

  dbs.Close

Exit_Comando1_Click:
  Exit Sub
```

```
Err_Comando1_Click:
   MsgBox Err.Description
   Resume Exit_Comando1_Click
End Sub



Option Compare Database

Public Sub ContatorePrimi()
   On Error GoTo Err_ContatorePrimi

   Dim dbs As DAO.Database
   Set dbs = CurrentDb
   Dim Record_set1 As DAO.Recordset
   Dim Record_set2 As DAO.Recordset
   Dim Record_set3 As DAO.Recordset
   Dim Primo As Long
   Dim UP As Long
   Dim NP_2 As Long
   Dim Controllo As Double

   UP = 0
   Primo = 0

   Set Record_set1 = dbs.OpenRecordset("SELECT primo FROM Primi _
         ORDER BY primo;", dbOpenDynaset, dbSeeChanges, dbOptimistic)

   If Record_set1.RecordCount > 0 Then
      Record_set1.MoveLast
      UP = Record_set1!Primo
      Record_set1.MoveFirst

      Set Record_set2 = dbs.OpenRecordset("SELECT primo FROM Primi _
            ORDER BY primo;", dbOpenDynaset, dbSeeChanges, dbOptimistic)
      Record_set2.MoveFirst
      NP_2 = Record_set2!Primo

      Do Until Record_set2.EOF
         Do Until Record_set1.EOF
            Primo = UP + (NP_2 + Record_set1!Primo)

            Set Record_set3 = dbs.OpenRecordset("SELECT primo FROM Primi _
                  WHERE primo <> 1 ORDER BY primo;", dbOpenDynaset, _
                                                dbSeeChanges, dbOptimistic)
```

```
            If Record_set3.RecordCount > 0 Then
               Record_set3.MoveFirst
               Do Until Record_set3.EOF
                  Controllo = Primo Mod Record_set3!Primo
                  If Controllo = 0 Then
                     GoTo SaltaPrimo
                  End If
                  Record_set3.MoveNext
               Loop
               Record_set3.Close

               If Not Controlla_Primo_Minore(Primo, UP) Then
                  dbs.Execute "INSERT INTO Primi (Primo) VALUES _
                                          (" & Primo & ")", dbSeeChanges
                  dbs.Execute "INSERT INTO UltimiPrimi (Primo_Trovato, _
                                       Ultimo_Primo, NP_1, NP_2) VALUES _
                                  (" & Primo & ", " & UP & ", " & NP_2 & ", " _
                                     & Record_set1!Primo & ")", dbSeeChanges
               End If
               GoTo esci
            End If
SaltaPrimo:
            Record_set1.MoveNext
         Loop
         Record_set1.MoveFirst
         Record_set2.MoveNext
         If Not Record_set2.EOF Then NP_2 = Record_set2!Primo
      Loop

      Record_set2.Close
      Record_set1.Close
   End If

esci:
   dbs.Close
Exit_ContatorePrimi:
   Exit Sub

Err_ContatorePrimi:
   MsgBox Err.Description
   Resume Exit_ContatorePrimi
End Sub
```

```
Public Function Controlla_Primo_Minore(ByVal Primo As Long, ByVal UP As
Long) As Boolean
   On Error GoTo Err_Controlla_Primo_Minore

   Dim dbs As DAO.Database
   Set dbs = CurrentDb
   Dim Record_set_4 As DAO.Recordset
   Dim Controllo As Double
   Dim I As Long
   Dim K As Long

   Controlla_Primo_Minore = False
   K = UP + 2

   If K = Primo Then GoTo esci_Controlla_Primo_Minore

   For I = K To Primo Step 2
      Set Record_set_4 = dbs.OpenRecordset("SELECT primo _
       FROM Primi WHERE primo <> 1 ORDER BY primo;", _
        dbOpenDynaset, dbSeeChanges, dbOptimistic)

      If Record_set_4.RecordCount > 0 Then
         Record_set_4.MoveFirst
         Do Until Record_set_4.EOF
            Controllo = I Mod Record_set_4!Primo
            If Controllo = 0 Then
               GoTo SaltaControllo
            End If
            Record_set_4.MoveNext
         Loop

         If I = Primo Then GoTo esci_Controlla_Primo_Minore

         If Not Verifica_Dispari(I, UP) Then
            Controlla_Primo_Minore = True
            GoTo esci_Controlla_Primo_Minore
         End If
SaltaControllo:
         Record_set_4.Close
      End If
   Next

esci_Controlla_Primo_Minore:
   dbs.Close
Exit_Controlla_Primo_Minore:
```

```
    Exit Function

Err_Controlla_Primo_Minore:
    MsgBox Err.Description
    Resume Exit_Controlla_Primo_Minore
End Function

Public Function Verifica_Dispari(ByVal Primo As Long, _
                                 ByVal UP As Long) As Boolean
    On Error GoTo Err_Verifica_Dispari

    Dim dbs As DAO.Database
    Set dbs = CurrentDb
    Dim Record_set_5 As DAO.Recordset
    Dim Record_set_6 As DAO.Recordset
    Dim N_pari As Long
    Dim NP As Long
    Dim ND As Long

    Verifica_Dispari = True
    N_pari = Primo - UP

    Set Record_set_5 = dbs.OpenRecordset("SELECT primo FROM Primi _
            ORDER BY primo;", dbOpenDynaset, dbSeeChanges, dbOptimistic)
    Set Record_set_6 = dbs.OpenRecordset("SELECT primo FROM Primi _
           ORDER BY primo;", dbOpenDynaset, dbSeeChanges, dbOptimistic)

    If Record_set_5.RecordCount > 0 Then
       Record_set_5.MoveFirst
       Record_set_6.MoveFirst
       NP = Record_set_6!Primo

       Do Until Record_set_6.EOF
          Do Until Record_set_5.EOF
             ND = N_pari - (NP + Record_set_5!Primo)
             If ND = 0 Then
                Verifica_Dispari = False
                dbs.Execute "INSERT INTO Primi (Primo) VALUES _
                                        (" & Primo & ")", dbSeeChanges
                dbs.Execute "INSERT INTO UltimiPrimi (Primo_Trovato, _
                                    Ultimo_Primo, NP_1, NP_2) VALUES _
                                  (" & Primo & ", " & UP & ", " & NP & ", " _
                                   & Record_set_5!Primo & ")", dbSeeChanges
                GoTo esci_Verifica_Dispari
             End If
```

```
            Record_set_5.MoveNext
        Loop
        Record_set_5.MoveFirst
        Record_set_6.MoveNext
        If Not Record_set_6.EOF Then NP = Record_set_6!Primo
      Loop

      Record_set_6.Close
      Record_set_5.Close
   End If

esci_Verifica_Dispari:
   dbs.Close
Exit_Verifica_Dispari:
   Exit Function

Err_Verifica_Dispari:
   MsgBox Err.Description
   Resume Exit_Verifica_Dispari
End Function
```

5.2.2 Python code for data sample analysis

For each result obtained, the corresponding code is shown below.

Global Frequency of Pairs (b, c)

This code reads the file, extracts columns b and c, combines them into pairs, and counts how many times each pair appears.

```
#From promp: pip install fsspec
import pandas as pd
from collections import Counter

def frequenza_coppie(file_path="testFile.txt"):
   try:
      df = pd.read_csv(file_path, sep='\s+', header=None)
      b = df.iloc[:, 2].tolist()
      c = df.iloc[:, 3].tolist()
      coppie = list(zip(b, c))
      conteggio_coppie = Counter(coppie)
      return conteggio_coppie
   except FileNotFoundError:
      return "Error: File not found."

conteggio_coppie = frequenza_coppie()
if isinstance(conteggio_coppie, Counter):
```

```
    print("Frequenza Globale delle Coppie (b, c):")
    for coppia, frequenza in conteggio_coppie.items():
        print(f"  pair {coppia}: {frequenza} frequency ")
else:
    print(conteggio_coppie)
```

Local Density of Pairs (b, c) in Windows

This code divides the data into windows of 100 rows and calculates the frequency of pairs within each window.

```
#From promp: pip install fsspec
import pandas as pd
from collections import Counter


def densita_coppie_finestre(file_path="testFile.txt", dimensione_finestra=100):
    try:
        df = pd.read_csv(file_path, sep='\s+', header=None)
        b = df.iloc[:, 2].tolist()
        c = df.iloc[:, 3].tolist()
        coppie = list(zip(b, c))
        risultati_finestre = []
        for i in range(0, len(coppie), dimensione_finestra):
            finestra = coppie[i:i + dimensione_finestra]
            conteggio_finestra = Counter(finestra)
            risultati_finestre.append(conteggio_finestra)
        return risultati_finestre
    except FileNotFoundError:
        return "Error: File Not Found."

risultati_finestre = densita_coppie_finestre()

if isinstance(risultati_finestre, list):
    print("\nLocal the pair density (b, c) in window:")
    for i, conteggio_finestra in enumerate(risultati_finestre):
        print(f"  Finestra da {i * 100} a {(i + 1) * 100}:")
        for coppia, frequenza in conteggio_finestra.items():
            print(f"  pair {coppia}: {frequenza} frequency ")
else:
    print(risultati_finestre)
```

Correlation between Prime Distance and b/c

This code calculates the distance between consecutive prime numbers and then the correlation between this distance and the values of b and c.

```
#From promp: pip install fsspec
import pandas as pd
import numpy as np
```

```
def preparaArray(file_content):
    array = []
    s = file_content.split("\n")

    for riga in s:
        if riga != '':
            vector = riga.split(" ")
            if len(vector) >= 4:
                p = vector[1].replace(',', '.')
                b = vector[2].replace(',', '.')
                c = vector[3].replace(',', '.')
                array.append([float(p), float(b), float(c)])
            else:
                print(f"Ignored line (less than 4 items): {riga}")
    return array

def calcola_distanza(p_values):
    return [p_values[i+1] - p_values[i] for i in range(len(p_values) - 1)]

def calcola_correlazione(distanze, b_values, c_values):
    if len(distanze) < 1 or len(b_values) < 1 or len(c_values) < 1 or \
       np.var(distanze) == 0 or np.var(b_values) == 0 or np.var(c_values) == 0:
        return None
    return np.corrcoef([distanze, b_values, c_values], rowvar=False)
# rowvar=False!

def elabora_dati(file_path="testFile.txt"):

    try:
        with open(file_path, 'r') as file:
            file_content = file.read()

        data_array = preparaArray(file_content)

        df = pd.DataFrame(data_array)

        p_values = df.iloc[:, 0].tolist()  # Colonna 0 (b) come p (??)
        b_values = df.iloc[:, 1].tolist()  # Colonna 1 (c) come b (??)
        c_values = df.iloc[:, 2].tolist()  # Colonna 2 (d) come c (??)

        tutte_le_correlazioni = []

        distanze = calcola_distanza(p_values)

        if len(distanze) > 0:
            correlazione = calcola_correlazione(distanze, b_values[:-1], c_values[:-1])
            if correlazione is not None:
                tutte_le_correlazioni.append(correlazione)

        return tutte_le_correlazioni

    except FileNotFoundError:
        return "Error: File Not Found."
    except Exception as e:
        return f"Error while processing: {e}"

risultati_correlazione = elabora_dati()
```

```
if isinstance(risultati_correlazione, list):
    print("\nCalculated correlations:")
    for correlazione in risultati_correlazione:
        print(correlazione)
else:
    print(risultati_correlazione)
```

Recursion and Frequency Analysis

This code counts the frequency of each value in columns b and c to identify recursion.

```
# From prompt: pip install fsspec
import pandas as pd
import numpy as np
from collections import Counter  # Importa la classe Counter

def analisi_ricorsione_frequenze(file_path="testFile.txt"):
    try:
        df = pd.read_csv(file_path, sep='\s+', header=None)
        b_values = df.iloc[:, 2].tolist()
        c_values = df.iloc[:, 3].tolist()

        conteggio_b = Counter(b_values)
        conteggio_c = Counter(c_values)
        return conteggio_b, conteggio_c
    except FileNotFoundError:
        return "Error: File Not Found"

conteggio_b, conteggio_c = analisi_ricorsione_frequenze()

print("Frequency of values in b:")
for valore, frequenza in conteggio_b.items():
    print(f"Value: {valore}, Frequency: {frequenza}")

print("\n Frequency of values in c:")
for valore, frequenza in conteggio_c.items():
    print(f"Value: {valore}, Frequency: {frequenza}")
```

Sinusoidal Rhythm Reconducibility (Fourier Transform)

This code performs the Fourier Transform on columns b and c to analyze their frequency components and displays the power spectra.

```
# From prompt: pip install fsspec
import pandas as pd
import numpy as np
from collections import Counter  # Importa la classe Counter

from scipy.fft import fft, fftfreq
```

```
import matplotlib.pyplot as plt

def preparaArray(file_content):
    array = []
    s = file_content.split("\n")

    for riga in s:
        if riga != '':
            vector = riga.split(" ")
            if len(vector) >= 4:
                p = vector[1].replace(',', '.')
                b = vector[2].replace(',', '.')
                c = vector[3].replace(',', '.')
                array.append([float(p), float(b), float(c)])
            else:
                print(f"Ignored line (less than 4 items): {riga}")  # Messaggio di debug
    return array


def analisi_fourier(file_path="testFile.txt"):
    try:
        with open(file_path, 'r') as file:
            file_content = file.read()

        data_array = preparaArray(file_content)

        df = pd.DataFrame(data_array)

        b_values = df.iloc[:, 1].tolist()
        c_values = df.iloc[:, 2].tolist()

        fft_b = fft(b_values)
        fft_c = fft(c_values)

        freq_b = fftfreq(len(fft_b))
        freq_c = fftfreq(len(fft_c))

        potenza_b = np.abs(fft_b)**2
        potenza_c = np.abs(fft_c)**2

        return freq_b, potenza_b, freq_c, potenza_c

    except FileNotFoundError:
        return "Error: File Not Found!"
    except Exception as e:
        return f"Error while processing: {e}"

freq_b, potenza_b, freq_c, potenza_c = analisi_fourier()

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.plot(freq_b, potenza_b)
plt.title('Spettro di Potenza di b')
plt.xlabel('Frequenza')
plt.ylabel('Potenza')

plt.subplot(1, 2, 2)
```

```
plt.plot(freq_c, potenza_c)
plt.title('Spettro di Potenza di c')
plt.xlabel('Frequenza')
plt.ylabel('Potenza')

plt.tight_layout()
plt.show()
```

Analysis of the distribution Δp = b + c compared with the Wigner-Dyson law

For a real comparison with the Wigner-Dyson distribution, an appropriate theoretical distribution must be generated. The Wigner-Dyson distribution is specific to Hermitian random matrices and depends on parameters such as the symmetry of the system.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import rv_histogram
from collections import Counter  # Importa la classe Counter


def preparaArray(file_content):
    array = []
    s = file_content.split("\n")

    for riga in s:
        if riga != '':
            vector = riga.split(" ")
            if len(vector) >= 4:
                p = vector[1].replace(',', '.')
                b = vector[2].replace(',', '.')
                c = vector[3].replace(',', '.')
                array.append([float(p), float(b), float(c)])
            else:
                print(f"Ignored line (less than 4 items): {riga}")
    return array


def analizza_distribuzione_differenze(file_path="testFile.txt"):

    try:
        with open(file_path, 'r') as file:
            file_content = file.read()

        data_array = preparaArray(file_content)

        df = pd.DataFrame(data_array)

        b = df.iloc[:, 1].values
        c = df.iloc[:, 2].values
        delta_p = b + c
```

```
        hist, bin_edges = np.histogram(delta_p, bins='auto', density=True)

        bin_mids = (bin_edges[:-1] + bin_edges[1:]) / 2

        return hist, bin_mids, bin_edges

    except FileNotFoundError:
        return None, None, None


hist, bin_mids, bin_edges = analizza_distribuzione_differenze()

if hist is None:
    print("Error: Cannot find file or error while parsing.")
    exit()

print("Valore di hist:", hist)
print("Valore di bin_mids:", bin_mids)
print("Valore di bin_edges:", bin_edges)

if isinstance(hist, np.ndarray) and len(hist) > 0:
    if len(bin_edges) > 1:
        bar_width = bin_edges[1] - bin_edges[0]
    else:
        bar_width = 1.0

    plt.figure(figsize=(10, 6))
    plt.bar(bin_mids, hist, width=bar_width)
    plt.title('Distribuzione di Δp = b + c')
    plt.xlabel('Valore di Δp')
    plt.ylabel('Densità di Probabilità')
    plt.show()

    rng = np.random.default_rng()
    wigner_dyson_sample = rng.normal(size=1000)
    wd_hist, wd_bin_edges = np.histogram(wigner_dyson_sample, bins='auto',
                                         density=True)
    wd_bin_mids = (wd_bin_edges[:-1] + wd_bin_edges[1:]) / 2

    # Visualizza la distribuzione di confronto
    plt.figure(figsize=(10, 6))
    if len(wd_bin_edges) > 1:
        wd_bar_width = wd_bin_edges[1] - wd_bin_edges[0]
    else:
        wd_bar_width = 1.0
    plt.bar(wd_bin_mids, wd_hist, width=wd_bar_width, alpha=0.5,
                                    label='Wigner-Dyson (Esempio)')
    plt.title('Confronto con Wigner-Dyson')
    plt.xlabel('Valore')
    plt.ylabel('Densità di Probabilità')
    plt.legend()
    plt.show()

else:
    print("Errore durante l'analisi della distribuzione. “ +
                            ”Impossibile visualizzare  l'istogramma.")
    print(delta_p)  # Stampa l'errore o i dati grezzi, se disponibili
```

Construction of the Cumulative Spectrum

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

def calcola_spettro_cumulativo(file_path="testFile.txt"):
    try:
        df = pd.read_csv(file_path, sep='\s+', header=None)
        b = df.iloc[:, 2].values
        c = df.iloc[:, 3].values
        delta_p = b + c

        delta_p_ordinato = np.sort(delta_p)

        cumulativa = np.arange(1, len(delta_p_ordinato) + 1)

        return delta_p_ordinato, cumulativa

    except FileNotFoundError:
        return "Error: File Not Found."

delta_p_ordinato, cumulativa = calcola_spettro_cumulativo()

if isinstance(delta_p_ordinato, np.ndarray):
    plt.figure(figsize=(10, 6))
    plt.plot(delta_p_ordinato, cumulativa)
    plt.title('Spettro Cumulativo di Δp = b + c')
    plt.xlabel('Valore di Δp')
    plt.ylabel('Numero Cumulativo di Occorrenze')
    plt.grid(True)
    plt.show()
else:
    print(delta_p_ordinato)
```

5.2.3 Python Application Prime Numbers Plugin for Blender

This addon for Blender 4.0+, titled: "Python Application Prime Numbers", is designed to import and visualize numerical data, especially related to numerical sequences (such as prime numbers), and transform them into dynamic 3D meshes. The main goal is to provide an interactive interface that allows the user to explore, connect and analyze numerical data through three-dimensional geometric structures.

How It Works Technically

• The code defines a main class ultimoPrimo that handles all operations.

• Numerical data is read from a .txt file, processed and converted into lists of vertices, edges and faces.
• Meshes are dynamically created using bpy.data.meshes.new() and linked to the active scene.
• Some functions perform operations on local files, such as os.path.exists() and os.remove(), ensuring a clean working environment.

Periodic Function Visualization

Two options ("Periodic function NP1" and "NP2") analyze selected columns of data through the Fourier transform to highlight periodic behaviors.
A plot of the sequence is also automatically generated with matplotlib. Furthermore, the plugin includes a context pie menu and a series of buttons accessible from the UI, making the user experience smoother and faster.

Code Python per Blender:

```
bl_info = {
    "name": "Python Application Numeri Primi",
    "author": "Douglas Ruffini",
    "version": (1, 0),
    "blender": (4, 0, 0),
    "location": "View3D > Python Application Numeri Primi",
    "category": "Add Mesh",
}


import bpy
from bpy.props import BoolProperty


class EditorSwitcherMenu(bpy.types.Menu):
    #Shortcut menu
    #Menu a scelta rapida
    bl_idname = "editor_switcher_pie_menu"
    bl_label = "Numeri Primi"

    def draw(self, context):
        pie = self.layout.menu_pie()
        pie.operator("object.import_numeri_primi",
        text="Import vertex",icon="VIEW3D").activate ="1"
        pie.operator("object.import_numeri_primi",
        text="Join vertex to edges",icon="NODETREE").activate ="2"
        pie.operator("object.import_numeri_primi",
        text="Join vertex to faces",icon="ACTION").activate ="3"
        pie.operator("object.import_numeri_primi",
        text="Vertex y & x long",icon="ACTION").activate ="5"
        pie.operator("object.import_numeri_primi",
```

```
        text="Vertex y & xz long",icon="ACTION").activate ="6"
        pie.operator("object.import_numeri_primi",
        text="Remuve vertex",icon="ACTION").activate ="4"
        pie.operator("object.import_numeri_primi",
        text="Periodic function  NP1",icon="ACTION").activate ="7"
        pie.operator("object.import_numeri_primi",
        text="Periodic function NP2",icon="ACTION").activate ="8"

class ultimoPrimo(bpy.types.Operator):

    bl_idname = "object.import_numeri_primi"

    bl_label = "Import vertex"
    bl_label_two = "Join vertex to edges"
    bl_label_three = "Join vertex to faces"
    bl_label_four = "Remuve vertex"
    bl_label_five = "Vertex y & x long"
    bl_label_six = "Vertex y & xz long"
    bl_label_seven = "Periodic function NP1"
    bl_label_eight = "Periodic function NP2"

    hl_label = "Import vertex"
    hl_label_two = "Join vertex to edges"
    hl_label_three = "Join vertex to faces"
    hl_label_four = "Remuve vertex"
    hl_label_five = "Vertex y & x long"
    hl_label_six = "Vertex y & xz long"
    hl_label_seven = "Periodic function NP1"
    hl_label_eight = "Periodic function NP2"

    bl_space_type = 'VIEW_3D'
    bl_region_type = 'UI'
    bl_category = "Scalar or Vector Panel"
    bl_options = {'REGISTER', 'UNDO'}


    global verts_b
    verts_b = []

    global verts
    verts = []

    global edges
    edges = []

    global faces
    faces = []

    global mesh
    mesh = None

    global obj
    obj = None

    global file_name
    file_name = ""

    global ind
```

```
ind =0

global index
index =0

global array
array= []

#Set restrictions on the file dialog
#Impostare le restrizioni alla finestra di dialogo file
filter_glob: bpy.props.StringProperty(default="*.txt", options = {'HIDDEN'})
filepath: bpy.props.StringProperty(subtype="FILE_PATH")
some_boolean: BoolProperty( name='Do a thing',
description='Do a thing with the file you\'ve selected', default=True, )

#Variable linked to the click of the object menu buttons
#Variabile legata al click dei pulsanti menu oggetto
activate: bpy.props.StringProperty(name="activate", description="activate")


#------------------------
#Start your code function
#Inizio delle tue funzioni

#Apri
def apri_file(nome_file, s):
    f = open(nome_file, s)
    return f

#Leggi
def leggi_file(f):
    return f.read()

def leggi_parte_file(f, n):
    return f.read(n)

def leggi_una_riga(f):
    return f.readline()

#Esiste
def esiste(path, os):
    try:
        return os.path.exists(path)

    except Exception as e:
        print("L'errore e' : ", e)

def esiste_file(sorgente, os):
    try:
        #Verifica la registrazione del path file
        #nomefile = os.environ.get(sorgente)
        #if nomefile and os.path.isfile(nomefile):
        return os.path.isfile(sorgente)

    except Exception as e:
        print("L'errore e' : ", e)

def esiste_dir(direc, os):
```

```
    try:
        #Verifica la registrazione della directory
        #nomedir = os.environ.get(direc)
        #if nomedir and os.path.isdir(nomedir):
        return os.path.isdir(direc)

    except Exception as e:
        print("L'errore e' : ", e)

#Elimina
def elimina_file(f, os):
    bool_=False
    try:
        os.remove(f)
        bool_ = True
        return bool_

    except Exception as e:
        print("L'errore e' : ", e)
        return bool_

def elimina_dir(c, os):
    bool_=False
    try:
        os.rmdir(c)
        return bool_

    except Exception as e:
        print("L'errore e' : ", e)
        return bool_

#Stampa
def stampa_file(t):
    print(t)

#Chiudi
def chiudi_file(f):
    try:
        f.close()

    except Exception as e:
        print("L'errore e' : ", e)

def preparaArray(f):

    global array
    array=[]
    s =f.split("\n")

    for riga in s:
        if riga !='':
            vector = riga.split(" ")
            a = ((vector[0]).replace(',', '.'))
            b = ((vector[1]).replace(',', '.'))
            c = ((vector[2]).replace(',', '.'))
            d = ((vector[3]).replace(',', '.'))
            # Consider columns 2, 3, 4
            array.append([float(b), float(c), float(d)])
```

```
    return array

def vertici(array):
    v = []

    for x in (array):
        v.append(x)

    return v

def lati(array):
    global edges
    edges = []
    edges = [[i, i+1] for i in range(len(array)-1)]

    return edges

def facce_C(array, k):
    global verts_b
    global faces

    verts_b = []
    faces = []
    h=0
    indice = 0
    linea = 0

    for i in array:
        if h<=k:
            linea = linea + i[1]

            verts_b.append( (linea, 0, 0) )
            verts_b.append( (0, i[1], 0) )
            verts_b.append( (0, 0, linea) )

            faces.append([indice, indice+1, indice+2])
            indice = indice + 3

        h = h+1

    return faces

def facce_B(array, k):
    global verts_b
    global faces

    verts_b = []
    faces = []
    h=0
    indice = 0
    linea = 0

    for i in array:
        if h<=k:
            linea = linea + i[1]

            verts_b.append( (linea, 0, 0) )
```

```
            verts_b.append( (0, i[1], 0) )
            verts_b.append( (0, 0, 0) )

            faces.append([indice, indice+1, indice+2])
            indice = indice + 3

        h = h+1

    return faces

def facce(array, k):
    global verts_b
    global faces

    verts_b = []
    faces = []
    h=0
    indice = 0

    for i in array:
        if h<=k:
            verts_b.append( (i[0], 0, 0) )
            verts_b.append( (0, i[1], 0) )
            verts_b.append( (0, 0, i[2]) )

        faces.append([indice, indice+1, indice+2])
        indice = indice + 3

        h = h+1

    return faces

def rimuoviMesh(bpy):
    #Removes the produced mesh
    #Rimuove la mesh prodotta
    global obj
    global file_name
    global array
    array=[]


    #if obj is not None:
    #obj_data = obj.data
        #Remuve object
        #Rimuovo l'oggetto
    #bpy.data.objects.remove(obj)
        #Then its data
        #Anche i suoi dati
    #bpy.data.meshes.remove(obj_data)

    for o in bpy.data.objects:
        if o.type == 'MESH':
            str=o.name
            if str.find("Primi")!= -1:
                obj_data = o.data
                bpy.data.objects.remove(o)
                bpy.data.meshes.remove(obj_data)
```

```
    if bpy.data.collections.get("Collection"):
        collection_to_remove = bpy.data.collections.get('Collection')
        for object in collection_to_remove.objects:
            if (object.name !='Camera') or (object.name !='Light'):
                str=object.name
                if str.find("Primi")!= -1:
                    obj_data = object.data
                    bpy.data.objects.remove(object)
                    bpy.data.meshes.remove(obj_data)

    obj=None
    file_name=""


    return obj

#Find the periodic function
#Trova la funzioni periodica
def caricaFunzionePeriodica(bpy, array, os, k):
    import numpy as np
    import matplotlib.pyplot as plt

    global edges
    global verts_b
    global faces

    edges = []
    verts_b = []
    faces = []

    # We only use the 'Np1' column for simplicity in detecting the
    # periodicity.
    y_values = [item[k] for item in array]  # Colonna Np1

    if len(y_values) < 2:
        print("Dataset insufficiente per analizzare la periodicità.")
        return False

    # Fourier transform to analyze periodicity.
    y_fft = np.fft.fft(y_values)
    frequencies = np.fft.fftfreq(len(y_values))
    magnitudes = np.abs(y_fft)

    # Determine the dominant frequency (ignore the zero frequency)
    dominant_freq_idx = np.argmax(magnitudes[1:]) + 1
    dominant_freq = frequencies[dominant_freq_idx]

    # Checking the periodicity based on the dominant frequency.
    if dominant_freq != 0:
        period = abs(1 / dominant_freq)
        print(f"Frequenza dominante trovata: {dominant_freq:.4f},” +
        “ periodo stimato: {period:.4f}")
        print("I dati mostrano un comportamento periodico.")
    else:
        print("Non è stata rilevata periodicità nei dati.")

    # Step 1: Generazione del Grafico e Salvataggio
    #image_path = '/tmp/sequenza_np1_grafico.png'
```

```
    # Specify a path for the image
    # Replace with your preferred path
    output_dir = "C:/path/to/output/"
    # Create directory if it doesn't exist
    os.makedirs(output_dir, exist_ok=True)
    image_path = os.path.join(output_dir, "sequenza_np1_grafico.png")

    plt.figure(figsize=(12, 6))
    plt.plot(y_values, marker='o', linestyle='-', color='b')
    plt.title("Sequenza di valori in 'y'")
    plt.xlabel("Indice")
    plt.ylabel("Valore")
    plt.grid()
    plt.savefig(image_path)
    plt.close()


    verts_b = [(i, y, 0) for i, y in enumerate(y_values)]  # (x, y, z)
    edges = [(i, i + 1) for i in range(len(verts_b) - 1)]

    return True

#These four functions are used by the dictionary
#Queste quattro funzioni servono al dizionario

def azioneUno(self, bpy, os):
    global obj

    if(ind=='1'):
        if (file_name == ""):
            ultimoPrimo.rimuoviMesh(bpy)
            ultimoPrimo.associaArray(self, os)

        obj=ultimoPrimo.caricaMesh(bpy, ultimoPrimo.vertici(array), [], [])

    return True


def azioneDue(self, bpy, os):
    global obj

    if (ind == '2'):
        if(file_name == ""):
            ultimoPrimo.rimuoviMesh(bpy)
            ultimoPrimo.associaArray(self, os)

        obj=ultimoPrimo.caricaMesh(bpy, ultimoPrimo.vertici(array),
        ultimoPrimo.lati(array), [])

    return True


def azioneTre(self, bpy, os):
    global obj

    if (ind == '3'):
        if (file_name == ""):
            ultimoPrimo.rimuoviMesh(bpy)
```

```
            ultimoPrimo.associaArray(self, os)

        ultimoPrimo.facce(array, index)
        obj=ultimoPrimo.caricaMesh(bpy, verts_b, [], faces)

    return True


def azioneQuattro(bpy):
    global file_name

    if (ind == '4'):
        #If it exists then I remove object
        #Se esiste allora rimuovo oggetto
        if obj is not None:
            ultimoPrimo.rimuoviMesh(bpy)
        else:
            file_name=""


def azioneCinque(self, bpy, os):
    global obj

    if (ind == '5'):
        if (file_name == ""):
            ultimoPrimo.rimuoviMesh(bpy)
            ultimoPrimo.associaArray(self, os)

        ultimoPrimo.facce_B(array, index)
        obj=ultimoPrimo.caricaMesh(bpy, verts_b, [], faces)

    return True


def azioneSei(self, bpy, os):
    global obj

    if (ind == '6'):
        if (file_name == ""):
            ultimoPrimo.rimuoviMesh(bpy)
            ultimoPrimo.associaArray(self, os)

        ultimoPrimo.facce_C(array, index)
        obj=ultimoPrimo.caricaMesh(bpy, verts_b, [], faces)

    return True

def azioneSette(self, bpy, os):
    global obj

    if (ind == '7'):
        if(file_name == ""):
            ultimoPrimo.rimuoviMesh(bpy)
            ultimoPrimo.associaArray(self, os)

        ultimoPrimo.caricaFunzionePeriodica(bpy, array, os, 1)
        obj=ultimoPrimo.caricaMesh(bpy, verts_b, edges, [])
```

```
    return True

def azioneOtto(self, bpy, os):
    global obj

    if (ind == '8'):
        if(file_name == ""):
            ultimoPrimo.rimuoviMesh(bpy)
            ultimoPrimo.associaArray(self, os)

        ultimoPrimo.caricaFunzionePeriodica(bpy, array, os, 2)
        obj=ultimoPrimo.caricaMesh(bpy, verts_b, edges, [])

    return True

def caricaMesh(bpy, verts_, edges_, faces_ ):
    global mash
    mesh = None
    global obj
    obj = None

    #Associa la mesh all'oggetto

    mesh = bpy.data.meshes.new("Primi")
    obj = bpy.data.objects.new("Primi",mesh)
    mat = bpy.data.materials.new(name="NewMaterial")

    #Associa al materiale il colore Rosso
    mat.diffuse_color = (1.0, 0.0, 0.0, 1.0)

    #Aggiungi la proprietà all'oggetto
    obj.data.materials.append(mat)

    #Posiziona l'oggetto
    obj.location = (0,0,0)

    #Ritorna la collezione dell'oggetto
    col = bpy.data.collections.get("Collection")
    col.objects.link(obj)

    #Associa la collezione alla scena
    bpy.context.view_layer.objects.active = obj

    #Proietta facce e vertici
    mesh.from_pydata(verts_, edges_, faces_)

    return obj


def associaArray(self, os):
    global index
    global array
    global file_name

    index = 0
    array = []

    file_name = self.filepath
```

```
    if file_name!="" or file_name!='':
        #il file e' nella directory del progetto
        if  ultimoPrimo.esiste(file_name, os):

            f=ultimoPrimo.apri_file(file_name, "r")

            array= ultimoPrimo.preparaArray(ultimoPrimo.leggi_file( f ) )

            index = len(array)-1

            ultimoPrimo.chiudi_file(f)

#End your code function
#Fine delle tue funzioni
#These two functions (execute, ivoke) are part of the plugin structure
#Queste due funzioni (execute, ivoke) fanno parte della struttura del plugin

@classmethod
def poll(cls, context):
    return True

def execute(self, context):
    import bpy
    import os
    global ind

    azione ={'1' : ultimoPrimo.azioneUno(self, bpy, os),
    '2' : ultimoPrimo.azioneDue(self, bpy, os),
    '3' : ultimoPrimo.azioneTre(self, bpy, os),
    '4' : ultimoPrimo.azioneQuattro(bpy),
    '5' : ultimoPrimo.azioneCinque(self, bpy, os) ,
    '6' : ultimoPrimo.azioneSei(self, bpy, os),
    '7' : ultimoPrimo.azioneSette(self, bpy, os),
    '8' : ultimoPrimo.azioneOtto(self, bpy, os)}

    azione.get(ind)

    return {'FINISHED'}

def invoke(self, context, event):
    global file_name
    global ind

    #Check button number
    #Assegno numero pulsante

    ind = self.activate


    if (ind=='1' and  file_name == "") or (ind=='2' and file_name == "") \
    or (ind=='3' and file_name == "")  \
    or (ind=='5' and file_name == "") or (ind=='6' and file_name == "") \
    or (ind=='7' and file_name == "") or (ind=='8' and file_name == "") :
        #Opens the file dialog
        #Apre la finestra di dialogo file
        context.window_manager.fileselect_add(self)
    else:
        return self.execute(context)
```

```
        return {'RUNNING_MODAL'}


def menu_func(self, context):
    #This function create the voices in the menu object
    #Questa funzione crea il menu oggetti e aggiunta di pulsanti sull'interfaccia
utente
    self.layout.operator_context = 'INVOKE_DEFAULT'
    #Buttons present in the object menu  object
    #Pulsanti presenti nel menu oggetto
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.bl_label,
    icon="VIEW3D").activate = "1"
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.bl_label_two,
    icon="NODETREE").activate="2"
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.bl_label_three,
    icon="ACTION").activate="3"
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.bl_label_five,
    icon="ACTION").activate="5"
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.bl_label_six,
    icon="ACTION").activate="6"
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.bl_label_four,
    icon="ACTION").activate="4"
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.bl_label_seven,
    icon="ACTION").activate="7"
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.bl_label_eight,
    icon="ACTION").activate="8"

def menu_header(self, context):
    #This function create the buttons in the ui
    #Questa funzione crea i pulsanti sull'interfaccia utente
    #Buttons present in the in the UI
    #Pulsanti presenti presenti nella UI
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.hl_label,
    icon="VIEW3D").activate = "1"
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.hl_label_two,
    icon="NODETREE").activate="2"
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.hl_label_three,
    icon="ACTION").activate="3"
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.hl_label_five,
    icon="ACTION").activate="5"
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.hl_label_six,
    icon="ACTION").activate="6"
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.hl_label_four,
    icon="ACTION").activate="4"
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.hl_label_seven,
    icon="ACTION").activate="7"
    self.layout.operator(ultimoPrimo.bl_idname, text =ultimoPrimo.hl_label_eight,
    icon="ACTION").activate="8"

#Store keymaps
#Memorizza le mappe dei tasti
addon_keymaps = []


def register():
    #This function register the class
    #Questa funzione registra la classe
```

```
    #Handle the keymap
    #Gestisci la mappa dei tasti
    wm = bpy.context.window_manager

    #Note that in background mode (no GUI available), keyconfigs are not available
    #either,
    #so we have to check this to avoid nasty errors in background case.
    #Tieni presente che in modalità background (nessuna GUI disponibile),
    #nemmeno i keyconfig sono disponibili,
    #quindi dobbiamo verificarlo per evitare errori spiacevoli nel caso in background.
    kc = wm.keyconfigs.addon

    if kc:
        km = wm.keyconfigs.addon.keymaps.new(name =
        "Window",space_type='EMPTY', region_type='WINDOW')
        kmi = km.keymap_items.new("wm.call_menu_pie",
        type = "E",alt=True, value = "PRESS")
        kmi.properties.name = "editor_switcher_pie_menu"
        addon_keymaps.append(km)

    bpy.utils.register_class(ultimoPrimo)
    bpy.utils.register_class(EditorSwitcherMenu)
    #Append
    #Aggiungi dopo
    bpy.types.VIEW3D_MT_object.append(menu_func)
    #Prepend
    #Aggiungi prima
    bpy.types.VIEW3D_HT_header.prepend(menu_header)


def unregister():
    #This function unregister the class
    #Questa funzione deregistra la classe
    #Note: when unregistering, it's usually good practice to do it in reverse order you
    #registered.
    #Can avoid strange issues like keymap still referring to operators already
    #unregistered...
    #Nota: quando si annulla la registrazione, di solito è buona norma farlo
    #nell'ordine inverso a quello della registrazione.
    #Può evitare problemi strani come la mappatura dei tasti che si riferisce ancora a
    #operatori giù non registrati...
    #Handle the keymap
    #Gestisci la mappa dei tasti
    wm = bpy.context.window_manager
    for km in addon_keymaps:
        for kmi in km.keymap_items:
            km.keymap_items.remove(kmi)
        wm.keyconfigs.addon.keymaps.remove(km)
    addon_keymaps.clear()

    bpy.types.VIEW3D_MT_object.remove(menu_func)
    bpy.types.VIEW3D_HT_header.remove(menu_header)
    bpy.utils.unregister_class(EditorSwitcherMenu)
    bpy.utils.unregister_class(ultimoPrimo)


if __name__ == "__main__":
    register()
```

6. Bibliography

Ruffini, D. (2025), "Electron Approach Theory. A Damped Oscillation Model Based on Relativistic Effects and Space-Time Feedback", (https://doi.org/10.31219/osf.io/hwca8_v1).
(Ruffini, 2025)

Russell, B. (1945), "A History of Western Philosophy", George Allen & Unwin Press.
(Russell, 1945)

Löf, E. R. M. (1980), "Intuitionistic Type Theory" , Bibliopolis Press
(Löf, 1980)

Gödel, K. "Collected Works" - Volume I: Publications 1929-1936 (1986); Volume II: Publications 1938-1974 (1990); Volume III: Unpublished Essays and Lectures (1995); Volume IV: Correspondence A-G (2003); Volume V: Correspondence H-Z (2003); Oxford University Press.
(Gödel, 1986)

Rutherford, E. (1920), "Nuclear Constitution of Atoms". Bakerian Lecture, Royal Society.
(Rutherford, 1920)

Edwards, H. M. (2003), "Lectures on the Riemann Zeta Function", American Mathematical Society.
(Edwards, 2003)

Dyson, F. J. (1962). "Statistical Theory of the Energy Levels of Complex Systems". I, II, III. Journal of Mathematical Physics, 3(1), 140–175.
(Dyson, 1962)

Montgomery, H. L. (1973). "The pair correlation of zeros of the zeta function". Proceedings of Symposia in Pure Mathematics, 24, 181–193.
(Montgomery, 1973)

Odlyzko, A. M. (1987). "On the distribution of spacings between zeros of the zeta function". Mathematics of Computation, 48 (177), 273–308.
(Odlyzko, 1987)

Csáki, C., Graesser, M., Randall, L., & Terning, J. (2000). "Cosmology of brane models with radion stabilization". Physical Review D, 62(4), 045015.
(Csaki et al., 1999)

Mehta, M. L. (2004). "Random Matrices*"* (3ª ed.). Elsevier.
(Mehta, 2004)

Hale, J. K., & Verduyn Lunel, S. M. (1993). *"*Introduction to Functional Differential Equations*"*. Springer.
(Hale, et al., 1993)

Connes, A. (1994). **"***Noncommutative Geometry"*. Academic Press.
(Connes, 1994)